

# Meridian Ada 4.1

DOS Environment Library
User's Guide

Copyright© 1987, 1988, 1989, 1990 Meridian Software Systems, Inc. All rights reserved. No part of this manual may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise without prior written permission of Meridian Software Systems, Inc. Printed in the United States of America.

The statements in this document are not intended to create any warranty, express or implied, and specifications stated herein are subject to change without notice.

Meridian Ada, Meridian-Pascal, and Meridian-C are trademarks of Meridian Software Systems, Inc.

OS/286 is a trademark of ERGO Computing Solutions.

386/ix is a trademark of INTERACTIVE Systems Corporation.

DECstation 3100 and ULTRIX are registered trademarks of Digital Equipment Corporation.

IBM, IBM PC, OS/2 and PS/2 are registered trademarks of International Business Machines Corporation.

Intel is a registered trademark of Intel Corporation.

Macintosh and SANE are registered trademarks of Apple Computer Corporation.

MPW, MultiFinder and QuickDraw are trademarks of Apple Computer Corporation.

Microsoft, PC DOS, and MS-DOS are registered trademarks of Microsoft Corporation.

NFS and Sun 3 are registered trademarks of Sun Microsystems, Inc.

Except where explicitly noted, uses in this document of trade names and trademarks owned by other companies do not represent endorsement of or affiliation with Meridian Software Systems, Inc. or its products.

# **Table of Contents**

Chapter	1 Introduction	
1.1	Scope of This Document	
1.2	Document Organization	
Chantar	2 Installing the DOS Environment Library	
2.1	Installation Procedure	
2.1	Adjusting an Unusual Installation	
2.2	Creating Library Links	
2.5 2.4	Verifying Correct Installation	
Chanter	3 Package Absolute_Disk	
3.1	Procedure Read	
3.1	Procedure Write	
3.2	Procedure write	
Chanter	4 Package Box	
4.1	Type Part	
4.2	Type User_Definition	
4.3	Type Simple_Kind	
4.3 4.4	Procedure Draw	
	.4.1 Using Default Boxes	
-	.4.2 Using a Selected Character	
Chapter	6 Package Cursor	
6.1	Procedure Set Size	
6.2	Procedure Inhibit	
6.3	Procedure Move	
6.4	Procedure Get_Position	
6.5	Procedure Up	
6.6	Procedure Down	
6.7	Procedure Left	
6.8	Procedure Right	
Chapte	r 7 Package Directory	ě
7.1	Function Make	
7.2	Function Remove	
7.3	Function Change_To	
7.4	Procedure Current_Name	
Chapte	r 8 Package Disk	
8.1	Procedure Reset	
8.2	Function Set_Default	
83	Function Get Default	

#### **Contents**

8.4	Procedure Get_Allocation_Info	•
8.5	Procedure Set_Verification	20
8.6	Function Verification_Is_On	20
8.7	Procedure Get_Free_Space	25
8.8	Procedure Get_Free_Space	27
		2
Chapter	9 Package Disk_Types	29
Chapter	10 Package Equipment	31
10.1	Function List	31
		31
Chapter	11 Package Errors	33
11.1	Procedure Get_Extended_Info	34
Chantan	12 Bealings Ett. 10	
12.1	12 Package File_IO	37
12.1 12.2	Procedure Create	41
	Procedure Create_Temporary	42
	2.2.1 Using Pathname	42
12.3	2.2.2 Using the Current Directory	43
	Procedure Create_New	44
12.4	Procedure Open	45
12.5	Function Close	45
12.6	Function Cooked	46
12.7	Function Raw	46
12.8	Procedure Read	46
12.9	Procedure Write	47
12.10	Function Delete	48
12.11	Function Rename	48
12.12	Procedure Move_File_Pointer	49
12.13	Procedure Get_Attributes	50
12.14	Function Set_Attributes	51
12.15	Function Set_File_Time	51
12.16	Procedure Get_File_Time	51
12.17	Procedure Dup_FileHandle	52
12.18	Procedure Cdup_FileHandle	52
12.19	Procedure Find_First	52
12.20	Procedure Find_Next	53
Chapter	13 Package Interrupt	55
13.1	Procedure Vector	55 55
-511		33
Chapter	14 Package Memory	57
14.1	Function Make	58
14.2	Procedure Split	58
14.3	Procedure Allocate	58
14.4	Function Release	59
14.5	Procedure Modify	59
14 6	Function Installed	<i>5</i> 0

14.7	Function Used	60
Chanton	15 Package Port	61
	Function In_Word	61
15.1	Function In_Byte	61
15.2	Procedure Out_Word	62
15.3	Procedure Out_Byte	62
15.4		
Chapter	16 Package Program_Control	<b>63</b> 64
16.1	Procedure Quit	64
16.2	Function Resident Ouit	
16.3	Procedure Get_Return_Code	64
16.4	Function Break_Status	64
16.5	Procedure Set Break_Status	64
16.6	Function Segment_Prefix	64
16.7	Function Execute	65
16.8	Function Msdos	65
16.9	Procedure Get_Environment_Variable	66
		<b>15</b>
Chapter	17 Package Revision	67
17.1	Exception Incorrect_Dos_Version	67
17.2	Function Dos	67
1	7.2.1 Returning Complete Revision Number	67
1	7.2.2 Returning Major Revision Only	67
		69
	· 18 Package Time	70
18.1	Procedure Set	70
. 1	8.1.1 Setting Time of Day	70
1	8.1.2 Setting Date	71
18.2	Procedure Get	71
1	8.2.1 Getting Time of Day	
1	8.2.2 Getting the Date	72
1	8.2.3 Getting Both Date and Time	72
18.3	Function Image	73
Chante	r 19 Package Tty	75
19.1	Procedure Clear_Screen	76
19.2		76
	19.2.1 Put Character	76
	19.2.2 Put String	77
	19.2.3 Put String, Non-Color Attributes	77
	19.2.4 Put String, Color Attributes	77
	19.2.5 Put Character, Non-Color Attributes	77
	19.2.6 Put Character, Color Attributes	78
19.3		. 78
19.3	<del>-</del>	78
19.4		79
19.5		79

#### **Contents**

19.6.1 R	aw Get	79
19.6.2 G	et Character	
19.6.3 G	et String	79
17.0.5	et String	79
Chapter 20 Pack	age Video	81
20.1 Type Vi	ideo_Mode	84
20.2 Procedu	ure Set	84
20.3 Procedu	ure Get_Mode	84
20.4 Procedu	ure Get_Light_Pen	84
20.5 Procedu	ure Set_Active	85
	ure Scroll_Up	85
	ure Scroll_Down	85 86
	ine Clear Screen	
	ure Clear_Screen	86
	re Read_Char	86
	rite Write_Char	87
20.10.1 W	rite With Attributes	87
20.10.2 W	rite With Previous Attributes	88
20.11 Procedu	ure Set_Color_Palette	89
20.11.1 Te	ext or Graphics	89
20.11.2 CC	GA	89
20.12 Procedu	ure Write_Tty	90
20.12.1 Wi	rite To Current Display Page	90
20.12.2 Wi	rite Graphic Character	91
20.13 Procedu	re Write_Pixel	91
20.14 Function	n Read_Pixel	91
20.15 Procedu	re Write_Graphic_Char	92
Index		O.F.
	••••••••••••••	95

# Chapter 1 Introduction

The Meridian Ada DOS Environment Library is a set of Ada packages for use with the Meridian Ada\* compiler that provides an interface to PC-DOS version 2.1 or later. The MeridianAda DOS Environment Library allows you to use most of the PC-DOS system calls, plus screen management and other BIOS functions. Excluded system calls include those that are obsolete in the newer versions of PC-DOS or are designed for use only on PC networks.

Note that the BIOS interface functions are designed to operate with the standard IBM PC BIOS.

Some operations are appropriate only for DOS 3.0 or later; these are marked as such. Any attempt to use these operations on earlier versions of DOS causes an exception to be raised.

Some of the highlights of the MeridianAda DOS Environment Library include:

Package	Function
absolute_disk	Provides procedures to read and write on disk devices without regard to the file system structure imposed by PC-DOS.
box	Provides the procedures for drawing boxes on the text screen.
common_display_types	Contains type declarations for the various packages that handle display operations.
cursor	Provides operations to move a text mode cursor, to get the current cursor coordinates, and to alter the visibility or form of the cursor.
directory	Provides subprograms that create, delete, and change directories.
disk	Provides operations such as flushing file buffers, getting default disk drive ID, and returning information about the disk drive.
disk_types	Contains type declarations used by the disk and directory operation packages.
equipment	Provides a function that enumerates the system-recognized devices and facilities present on the machine.
errors	Provides PC-DOS status declarations and for version 3.0 or later of PC-DOS, a procedure to get extended status information.
file_io	Provides input/output, file attribute manipulation, and wildcard search operations.
interrupt	Allows calls to the interrupt vectors.
memory	Provides functions to allocate and deallocate memory.
port	Allows byte or word input and output to the specified port.
program_control	Provides functions that exert control over the executing program that execute other programs, or that obtain information about resident programs.

<sup>\*</sup>The DOS Environment Library can also be used with the AdaGraduate Compiler. Please note that references to the Meridian Ada Compiler also apply to the AdaGraduate Compiler.

Package	Function
revision	Provides a facility for determining the revision number of PC-DOS.
time	Provides procedures to get and set the current system date and time.
tty	Provides operations on the console terminal display and keyboard.
video	Provides various output and control functions for the Monochrome, Color Graphics and Extended Graphics Adapter Cards.

## 1.1 Scope of This Document

It is assumed that you are familiar with PC-DOS and the Meridian Ada compiler and associated library management tools. This document covers only material required to use the packages; information about the Ada programming language, about most low-level functional aspects of the PC-DOS system calls, and most aspects of the Meridian Ada compiler is outside the scope of this document. Other documents that cover the aforementioned material include:

- The Meridian Ada Compiler User's Guide. This document describes operation of the compiler and associated tools.
- Reference Manual for the Ada Programming Language ANSI/MIL-STD-1815A (the LRM). This document describes the Ada language.
- Ada For Programmers (Prentice-Hall, 1983). This book by Meridian co-founders
  Eric Olsen and Stephen Whitehill provides an introduction to Ada programming
  for practicing programmers.
- Microsoft MS-DOS Operating System Programmer's Reference (available from Microsoft Corporation). This document covers DOS system calls in great detail.

# 1.2 Document Organization

The descriptions of the packages are organized alphabetically by package name. Each package is presented with its specification, a brief description of its uses, and a discussion of the details of the package.

# Chapter 2 Installing the DOS Environment Library

#### 2.1 Installation Procedure

Prior to installing the Meridian Ada DOS Environment Library, the Ada compiler must be installed.

This installation procedure assumes that you will install the DOS Environment Library on the same hard disk drive and in the same top-level directory in which you installed the compiler. If you do not install the DOS Environment Library in that place (not recommended) the installation procedure will work to a point, but the link entry in the DOS Environment Library library database file (ada.lib) will have to be linked to the standard distribution library (paclib/ada.lib). An example of how to do this is given in section 2.2.

To perform the installation procedure:

- 1. Reboot the operating system.
- 2. Insert the distribution diskette into diskette drive a:.
- 3. Run the installation program as:
  - a:install d:\directory

The command line arguments to install are d, a hard disk drive letter, and directory. Note that directory must start with a backslash ("\"). The hard disk and directory that you select should be the same as were selected for installation of the compiler (e.g. c:\ada).

The installation procedure should take only a few minutes. If there is not enough space on the destination disk to accommodate all the files, the installation will fail. The files are installed in a sub—directory named dosenv below the top—level directory given to the install command. If dosenv does not exist on the destination disk, then the install command creates that directory. If the directory already exists, a harmless error message is printed, but installation proceeds.

# 2.2 Adjusting an Unusual Installation

If you did not install the DOS Environment Library using the same top-level directory in which you installed the compiler, you should have gotten a warning message. In this case, the link entry in the DOS Environment Library library database file (ada.lib) must to be modified to reflect the actual location of the standard distribution library (paclib\ada.lib).

You do not need to do this if you used the same top-level directory in which you installed the compiler.

The adjustment to the DOS Environment Library library is made by using the lnlib command, as in this example:

```
e:
cd \stuff\dosenv
rem -- The above two commands assume that e:\stuff was
rem -- the top-level directory specified to the install command.
lnlib -r c:\ada\paclib\ada.lib e:\ada\paclib\ada.lib
```

This example assumes that you originally installed the compiler using **e:\ada** as the installation directory and that you installed the *DOS Environment Library* in **e:\stuff**. Note that the **lnlib** command in this example replaces a pre-existing library link entry.

## 2.3 Creating Library Links

Once the software is installed, it is necessary to make the appropriate link in each local library where the package is to be used. If the software is installed in the same top-level directory in which the compiler was installed, then the installation procedure automatically modifies the behavior of the newlib command so that the DOS Environment Library library is linked into every newly created local library. If the software is installed elsewhere and you want the same modification made to the newlib command, then you should perform the following procedures.

Modify the newlib.bat file to link the DOS Environment Library library whenever the newlib command is invoked. For example, this command could be added to the end of the newlib.bat file:

```
lnlib c:\ada\dosenv\ada.lib
```

The newlib.bat file is located in the bin directory below the Meridian Ada compiler installation directory. In this example, it is assumed that c:\ada is the top-level directory where the Meridian Ada compiler was installed.

Note: Always make a backup of the old newlib.bat file before modifying it.

## 2.4 Verifying Correct Installation

Below the top—level directory given to install, the directory dosenv and test should have been created by the installation procedure for the DOS Environment Library. Note that if you installed the DOS Environment Library in the same top—level directory in which you originally installed the compiler (e.g. c:\ada) the test directory already existed.

There should be a number of files present in the dosenv directory: \*.lib, \*.aar, and \*.int.

In the test directory these two source files should be present:

boxdemo.ada

Draws boxes on the screen. In addition to demonstrating several of the DOS Environment packages, it also demonstrates Meridian Ada tasking.

NOTE: Because this program uses the screen management packages, it requires a 100% IBM—compatible BIOS.

envdisp.ada

Displays current environment settings. This prints the current directory, the names of files in the current directory, environment variable values, and other environment—specific information.

To compile and run these demonstration programs, follow these steps:

1. Go to the test directory and enter a link in the local library to the DOS Environment Library library, as in this example:

c: cd ada

cd ada/test

newlib

lnlib c:\ada\dosenv\ada.lib

rem -- If newlib has been modified to do this

rem -- already, you can skip the Inlib command.

This example assumes that the DOS Environment Library was installed using c:\ada as the top-level directory. Use whatever path is appropriate for your installation.

2. Compile and link the first sample program:

ada boxdemo.ada bamp boxdemo 3. Assuming that all went well, run the sample program as:

#### boxdemo

This should clear the display and draw some boxes. If it cannot do this with the current display mode, an error message is printed. Regardless of precisely what is displayed, something should happen.

4. Compile and link the second sample program:

ada envdisp.ada bamp envdisp

5. Assuming that all went well, run the second sample program as:

#### envdisp

This should print various information about the current environment, as detailed in the program description above. Regardless of precisely what is displayed, something should happen.

If these programs work correctly (within their operational parameters), then the DOS Environment Library was probably loaded correctly and is ready for further use in other programs.

#### Installation

# Chapter 3 Package Absolute\_Disk

Package absolute\_disk provides procedures to read and write on disk devices without regard to the file system structure imposed by PC-DOS.

#### **SPECIFICATION**

```
with disk_types,
     system;
package absolute disk is
  type status is (
                         bad command,
     ok,
      address_not_found, write_protected,
                         dma failure,
     sector_not_found,
                          controller failed,
     bad crc,
                         time out,
     seek failed,
     unknown error
    );
   procedure read
                                : disk types.drive_id;
      from drive
                               : natural;
     number of sectors
      starting logical sector : natural;
                               : system.address;
      transfer area
                                : out status
      error
    );
   procedure write (
      from drive
                                : disk types.drive id;
      number of sectors
                                : natural;
      starting logical sector : natural;
                                : system.address;
      transfer area
                                : out status
      error
    ) :
end absolute disk;
```

#### 3.1 Procedure Read

The procedure read gets data from specific logical disk sectors, ignoring the file system structure of the disk. The selected disk seeks to the starting\_logical\_sector, reads the number\_of\_sectors specified into the memory location starting at the transfer\_area, and returns an error status. If no error occurs, status' (ok) is returned. Any other status value indicates some kind of error.

#### Package absolute disk

The number of bytes read per sector is typically 512, but this number may vary according to the disk device type.

Logical sectors are numbered sequentially from track 0, head 0, sector 1, and continue until the last sector, the last sector number being a characteristic of a specific type of disk device. As logical sector numbers increase, the internal sector, head, and track numbers increment (in that order).

The **read** service works with logical drives as opposed to true physical drives. An attempt to read from a logical drive that has no corresponding physical drive or disk area (i.e is not installed) has unpredictable results.

This call corresponds to Interrupt 16#25#.

#### 3.2 Procedure Write

The procedure write puts data to specific logical disk sectors, ignoring the file system structure of the disk. The selected disk seeks to the starting\_logical\_sector, writes the number\_of\_sectors specified from the memory location starting at the transfer\_area, and returns an error status. If no error occurs, status' (ok) is returned. Any other status value indicates some kind of error.

The number of bytes written per sector is typically 512, but this number may vary according to the disk device type.

Logical sectors are numbered sequentially from track 0, head 0, sector 1, and continue until the last sector, the last sector number being a characteristic of a specific type of disk device. As logical sector numbers increase, the internal sector, head, and track numbers increment (in that order).

The write service works with logical drives as opposed to true physical drives. An attempt to write to a logical drive that has no corresponding physical drive or disk area (i.e is not installed) has unpredictable results.

\*\* WARNING \*\* Writing sectors directly to a disk in this way is likely to destroy any previous disk file structure.

This call corresponds to Interrupt 16#26#.

The package box provides procedures for drawing boxes on the text screen either using the box-drawing characters (tops, bottoms, sides, and corners) or using a single selected character.

The routines in package box call many DOS functions; there is no correspondence to a single DOS function as with most of the other subprograms in the *Meridian Ada DOS Environment Library*.

#### **SPECIFICATION**

```
with common display types;
use common display types;
package box is
  type part is
     north, northeast, east, southeast,
     south, southwest, west, northwest
     );
   type user definition is array (part) of extended_ascii;
   type simple kind is (
     single_sided, double_sided,
                  double top
     single top,
     );
procedure draw (
                      : row_range;
   upper_left_row
   upper_left_column : column range;
   lower_right_row
                      : row range;
   lower_right_column : column_range;
                       : simple kind := single_sided;
   kind
                       : display attribute := (
   attribute
      foreground => white,
     background => black,
     blink
              => false
     );
                       : display page
                                           := 0
   page
  );
```

```
procedure draw (
      upper_left_row
                             : row_range;
      upper left column : column range;
      lower right row
                             : row range;
      lower right column : column range;
                             : user definition;
      box definition
      attribute
                             : display attribute := (
         foreground => white,
         background => black,
         blink
                      => false
         );
                             : display page
      page
     );
   end box;
4.1
      Type Part
Type part is an enumeration used to describe the parts of a box.
   type part is (
         north, northeast, east, southeast,
         south, southwest, west, northwest
     );
The elements are compass points and have these correspondences:
      North
                   The top of a box
      NorthEast
                   The upper right corner of a box
      East
                   The right side of a box
      SouthEast
                   The lower right corner of a box
      South
                   The bottom of a box
                   The lower left corner of a box
      South West
                   The left side of a box
      West
      NorthWest
                   The upper left corner of a box
These are illustrated as:
                                       North
                                                             NorthEast
             NorthWest
                                                             East
                  West
                                                             SouthEast
             SouthWest
```

South

# 4.2 Type User\_Definition

Type user\_definition is used to describe what character is used for each part of a box.

```
type user definition is array (part) of extended ascii;
```

Some particularly useful characters for drawing boxes are in the range 179.. 218. If no table showing these characters is available, then the following program will print them out:

```
with video;
procedure printchars is
begin
   for char in 179 .. 218 loop
      video.write_tty (char); -- write to page 0
      video.write_tty (' '); -- separate with space
end loop;
end printchars;
```

# 4.3 Type Simple Kind

Type simple kind is used to select a pre-defined box style.

```
type simple_kind is (
    single_sided, double_sided,
    single_top, double_top
);
```

The default box-drawing characters are the IBM PC characters used to draw connecting horizontal and vertical lines and corners, with either single rules or double rules.

The box styles corresponding to each simple\_kind are:

```
single_sided single—rule lines all around
double_sided double—rule lines all around
single_top single—rule lines on top and bottom; double rule lines on the sides
double top double—rule lines on top and bottom; single rule lines on the sides
```

#### 4.4 Procedure Draw

There are two overloaded versions of procedure draw, the disambiguating parameter being kind or box\_definition. The first version draws a few kinds of boxes using default box-drawing characters, the second version uses selected characters for each part of a box.

## 4.4.1 Using Default Boxes

The first version of procedure draw displays one of several pre-defined box types using default box-drawing characters.

```
procedure draw (
  upper left row : row_range;
  upper_left_column : column_range;
  lower right row : row range;
  lower right column : column range;
                     : simple kind := single sided;
  kind
                     : display attribute := (
  attribute
     foreground => white,
     background => black,
     blink => false
    );
                     : display_page := 0
  page
  );
```

Given the upper left and lower right coordinates, a box is drawn on the specified display page with the specified style and attributes.

This call is valid for text modes only.

Invalid coordinates or attempting to exceed the current mode's display size has unpredictable results.

An example for drawing a single-sided box follows.

```
-- draw a single-sided box at (row/col)
     - 1
-- in page zero, with White/Black blinking attributes.
with common_display_types,
     box;
use common display_types;
procedure box test1 is
   attr : display attribute;
begin
   attr.blink := true;
  box.draw (
     upper left row
                        => 5,
     upper_left_column => 5,
     lower right row => 10,
     lower_right_column => 20,
                 => attr
     attribute
     );
end:
```

## 4.4.2 Using a Selected Character

The second version of procedure draw uses a selected character to draw each box part.

```
procedure draw (
   upper_left_row : row_range;
   upper_left_column : column_range;
   lower_right_row : row_range;
   lower_right_column : column_range;
   box_definition : user_definition;
   attribute : display_attribute := (
      foreground => white,
      background => black,
      blink => false
      );
   page : display_page := 0
):
```

Given the upper left and lower right coordinates, a box is drawn with the specified character and attributes in the specified display page.

This call is valid for text modes only.

Invalid coordinates or attempting to exceed the current mode's display size has unpredictable results.

An example of procedure draw using a selected character follows.

```
-- draw a box with the Happy Face character at (row/col)
-- in page zero, with Cyan/Black attributes.
with common_display_types,
     box:
use common display types;
procedure box_test2 is
  attr : display attribute;
  my box : box.user definition := (others => 1);
     -- 1 = Happy Face
begin
  attr.foreground := cyan;
  box.draw (
                        => 5,
     upper left row
     upper left column => 5,
     lower_right_row => 7,
     lower_right_column => 30,
     box_definition => my_box,
     attribute
                       => attr
    );
end:
```

Note that a different box-drawing character from the IBM PC character set may be used for each part.

## Package box

# Chapter 5 Package Common\_Display\_Types

The package common\_display\_types contains type declarations for the various packages that handle display operations: box, cursor, tty, and video.

#### SPECIFICATION

```
package common display types is
   subtype byte is integer range 0 .. 255;
   subtype extended ascii is integer range 0 .. 255;
   subtype column range
                            is integer range 0 .. 79;
                            is integer range 0 .. 24;
   subtype row range
   subtype cursor_size is integer range 0 .. 13;
   subtype display page is integer range 0 .. 7;
   type color is (
                                    green,
     black,
                   blue,
                   red,
                                    magenta,
     cyan,
     brown,
                   white,
                                    grey,
     light blue, light green,
                                    light cyan,
                   light magenta,
     light_red,
                                    yellow,
     bright white
     );
   subtype background color is color range black .. white;
   type display attribute is
     record
         foreground : color
                                       := white:
        background : background color := black;
                    : boolean
                                       := false;
        blink
      end record:
    - underline valid for monochrome mode only.
   underline : constant display attribute := (
      foreground => blue,
      background => black,
      blink
                => false
   subtype color palette is integer range 0 .. 1;
   type graphic color is (
      background, color1,
      color2,
                    color3
     );
end common display types;
```

# Charge of Endlyge Common Display Types

```
and the court out of the sections of the configuration being suttention being the property of property of the configuration of the conf
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               TO COLOR DE CONTRACTOR DE CONT
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               Windy at the late
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        the interpretation administration of the contraction of the contractio
                                                                                                                                                                                                                                                                                                                                                                                                                                en 1925 - O episka, koyenda al north egyptu e
                                                                                                                                                                                                                                                                                            substyle each each leader in the sage as a series of the contraction o
                                                                                                                                                                                                                                                                           The same and the s
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        apast janakan banggapar
                                                                                                                                                                                                                                                                           The same of the sa
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 Lagragier ach ach teles
                                                                                                                                                                                                                                                                                                                                                        still o opena repecal ex este mesus ecyclos
                                                                                                                                                                                                                                                                                                                                                                The the space of a late got stage to large the large of the talk got the same
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   ) Pi soluw Mest
                                                                                                                                                                                                                                                                                                                                                                                                                                                                       LE GREEN
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    1.0
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          A Buch
                                                                                                                                                                                                                                                                                                                                                                                                                            and a strain
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                92
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  3 14 164
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            10 T
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    January Land
                                                                                                                                                                                                                                                                                                                                                                       · mayor animal i
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 Constant Control
                                                                                                                                                                                                                                                                                                                                                                                                                                               light dagana, systlem
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          San San San
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        9.587
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   erin ingi d
                                                                                                                                                                                                             2 2 1 W
                                                                                                                                                                                                                                                                                                                                           ි එකමන්ත් පතුර මත් කත විතිය වැනි කට ලිනින්න සිට මෙදුලන විලික
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   ्वके अवस्थाय नेपन्यक्त । इस विकास
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      0 F 765
                                                                                                                                                                                                                                                                                                                                                                       Coletan : botton about
                                                                                                                                                                                                                                                                                                                                                        and the selection of the contract of the contr
                                                                                                                                                                                                                                                                                                                                                            ozolika ≠.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  restord a solida
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    Coronal F
                                                                                                                                                                                                                                                                                                                                                                       -yine bind waterdooren tet baler ookusebrup ---
                                                                                                                                                                                                                                                                                                                                                                                 o va o o o tinitata ya Tanàh taka son o osahin kenda.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               Landy Kar Carry Mandrid
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               Aprila in the product of
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  abbie – cellai palkata la intage e sanço C -
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               The state of the s
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            include annough said
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                ExoLog
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 35 34 4
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   charge prignit are to his
```

Package cursor provides operations to move a text mode cursor, to get the current cursor coordinates, and to alter the visibility or form of the cursor.

The Color/Graphics Adapter has eight display pages (0.. 7) in 40 column by 25 line text modes and four display pages in the 80 column by 25 line text modes. The Monochrome adapter has only one display page. Specifying an invalid display page (i.e. a display page that is not defined for the current text display mode) has unpredictable results.

#### **SPECIFICATION**

```
with
        common display types;
use
        common display types;
package cursor is
  procedure set size (
     start_line : cursor_size;
     end line : cursor size
   );
  procedure inhibit;
  procedure move (
     row
              : row range;
     column : column range;
              : display page := 0
     );
   procedure get position (
             : out row range;
     column : out column range;
     page
              : display page :=
     );
   procedure up
                    (page : display_page := 0);
   procedure down (page: display page := 0);
   procedure left (page: display_page := 0);
   procedure right (page: display page
end cursor;
```

# 6.1 Procedure Set Size

Procedure set size sets the start line and end\_line of the blinking cursor.

```
procedure set_size (
   start_line : cursor_size;
   end_line : cursor_size
);
```

This call is valid only in Text modes.

The ROM BIOS default values for start\_line and end\_line are:

text display mode	Start_Line	End_Line
Text80_BW_Ma (Mode 7)	11	12
Text40_BW (Mode 0)	6	7
Text40_CO (Mode 1)	6	7
Text80_BW (Mode 2)	6	7
Text80_CO (Mode 3)	6	7

A particular text display mode is selectable via the procedure **video.set**. Refer to package **video** for a description of the various text modes.

This call corresponds to Interrupt 16#10#, function 16#01#.

#### 6.2 Procedure Inhibit

Procedure inhibit inhibits display of the cursor (i.e. makes it invisible).

```
procedure inhibit;
```

This call is valid only in Text modes.

This call corresponds to Interrupt 16#10#, function 16#01#.

#### 6.3 Procedure Move

Procedure move moves the cursor to the specified row and column within the specified display page.

```
procedure move (
   row : row_range;
   column : column_range;
   page : display_page := 0
);
```

Moving the cursor to a location that is invalid for the current display mode has unpredictable results. The coordinate scheme is rectangular:

- (0,0) Coordinate (Row 0, Column 0) is the upper left corner of the display.
- (24,79) Coordinate (Row 24, Column 79) is the lower right corner of the display in 80-column text modes.

This call corresponds to Interrupt 16#10#, function 16#02#.

An example follows:

```
-- Move the cursor to row 10, column 20 in display page
-- 0 (default page).
--
cursor.move (row => 10, column => 20);
```

# 6.4 Procedure Get\_Position

Procedure get\_position returns the current cursor position for the specified display page.

```
procedure get_position (
   row : out row_range;
   column : out column_range;
   page : display_page := 0
);
```

This call corresponds to Interrupt 16#10#, function 16#03#. An example follows.

## 6.5 Procedure Up

Procedure up moves the cursor up one location in the specified display page.

```
procedure up (page : display_page := 0);
```

Attempts to move the cursor up past the top of the screen are ignored.

This call corresponds to Interrupt 16#10#, function 16#02#.

An example follows.

```
-- Move the display page zero cursor up one location. -- cursor.up;
```

#### 6.6 Procedure Down

Procedure down moves the cursor down one location in the specified display page.

```
procedure down (page : display_page := 0);
```

Attempts to move the cursor down past the bottom of the screen are ignored.

This call corresponds to Interrupt 16#10#, function 16#02#.

An example follows.

```
-- Move the display page one cursor down one location.
--
cursor.down (page => 1);
```

#### 6.7 Procedure Left

Procedure left moves the cursor left one location in the specified display page.

```
procedure left (page : display page := 0);
```

Attempts to move the cursor left past the edge of the screen are ignored.

This call corresponds to Interrupt 16#10#, function 16#02#.

# 6.8 Procedure Right

Procedure right moves the cursor right one location in the specified display page.

```
procedure right (page : display page := 0);
```

Attempts to move the cursor past the right edge of the screen are ignored.

This call corresponds to Interrupt 16#10#, function 16#02#.

```
That is the first and the second of the seco
                                             and the continues of the test of the continues of the con
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  I material at the con-
                                                                                                                                                                                                                                                                                                                                                                                                                                           STORE LANGETY CO.
                                                                                                                                                                                                                                                                                                                                                                                                                                        all weather and the
                                                                             The state of the second of the
                                                                                                                                                                                                                 THE HE SPACE WATER TO SPACE THE SECOND
                                                                                                                                                      . botanga सम्बद्धा अन्तरमञ्जूषा स्वीति सम्बद्धा कर्ता (ब्रिक्ट क्रिक्ट स्वाति सम्बद्धाः । अन्तर । अन्तर स्वाति
                                                                                                                                                                                                                                   Pland I diction as first three regions of constructions are sent
                                             and reach it to the term the transfer of the transfer of the contract of the c
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           "我们就是我的我"
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            The copy of the trailing of the contract of th
                                                                                                                                                                                       to an appear year and a specific and a contract of
                                                                                  Contract the deside and in include and their places from the contract of the end of
                                                                                                                                                                                                          years and an expensive or entertain the first the contract of the first
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           المناوي والمناوي والمنطقة والمناورات
                     the first a first through the state on some of the first sections
                                                                                                                                                                                                                                                                                                                                                                                                       (1) Company and the rest
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         and the second second consistent through the parties of the second second second second second second second se
                                                                                                                                                                   read the state of the terminal description of the state o
                                                              uble site use mosterial is subjected as the control of the control
                                                                                                                                                                                                              TOTAL TOROUGH MOTEL AND WALL OF PROPERTY IN THE TOTAL
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              李骥就事:
          atina ngalajan ng babagi na sai ing baasi na pagalaga nagalaga na tao ang natara ag masar ng masar ng mg
                                                                                                                                                             (C will report your arth of the part of the original
                                                                                                         รุ่มราชหาสู่รับทร คือวิทย์สุดคร (ปฏิบัติรู้มี) หรือที่ ใชิติ การกรายสะบัติสุด ขพายสาราส สาราสาราส
                                                                                                                                                                                                        The State of the Asset Asset and Asset as a state of the state of
```

# **Chapter 7 Package Directory**

end:

Package directory provides subprograms that operate on directories. A program using this package can create or delete directories and can change the current directory of the program.

```
SPECIFICATION
  with errors.
        disk types;
  package directory is
      subtype pathname is string (1 .. 65);
      function make (name : in string)
         return errors.extended errors;
      function remove (name : in string)
         return errors.extended errors;
     function change to (name : in string)
         return errors.extended errors;
     procedure current name (
         for drive : disk_types.drive_designator;
                    : out pathname;
         name
         last
                    : out natural;
         error
                    : out errors.extended errors
        );
   end directory;
7.1
     Function Make
Function make creates a directory.
   function make (name : in string) return
      errors.extended.errors;
This call corresponds to Interrupt 16#21#, function 16#39#.
Possible errors are:

    errors.path not found

      • errors.access denied
Examples which use function make follow.
   -- Create the directory TEST.DIR in the root directory
   -- of the 'C' Drive.
   with errors,
         directory;
   procedure dir test1 is
      error : errors.extended errors;
      error := directory.make (name => "C:\TEST.DIR");
```

Another example follows.

```
-- Create the directory TEST.DIR in the current directory
-- on the default disk drive.
--
with errors,
    directory;
procedure dir_test2 is
    error : errors.extended_errors;
begin
    error := directory.make (name => "TEST.DIR");
end;
```

#### 7.2 Function Remove

Function **remove** removes a directory.

```
function remove (name : in string) return
errors.extended_errors;
```

This call corresponds to Interrupt 16#21#, function 16#3A#.

Possible errors are:

- errors.path\_not\_found
- errors.access\_denied
- · errors.remove current directory

An example which uses function **remove** follows.

```
-- Remove TEST.DIR from the current drive and directory.
with directory,
errors;
procedure remove_test is
error : errors.extended_errors;
begin
error := directory.remove (name => "TEST.DIR");
end;
```

# 7.3 Function Change\_To

Changes the current directory for the program.

```
function change_to (name : in string)
  return errors.extended_errors;
```

This call corresponds to Interrupt 16#21#, function 16#3B#.

A possible error follows:

errors.path\_not\_found

An example of using package directory with the function change to follows.

```
-- Change the current working directory to the parent directory
  with directory,
         errors;
  procedure change test is
      error : errors.extended errors;
     error := directory.change to (name => "..");
   end:
7.4
     Procedure Current Name
Procedure current name obtains the pathname of the current working directory for the specified drive.
  procedure current name (
      for drive : disk types.drive designator;
                 : out pathname;
      name
                  : out natural;
      last
                 : out errors.extended errors
      error
     );
The returned pathname does not include the drive identifier.
The returned parameter last denotes the index of the last valid character position in name.
This call corresponds to Interrupt 16#21#, function 16#47#.
A possible error follows:
      • errors.invalid drive
An example of using package directory with procedure current name follows.
   -- Get and display the current directory for the 'A' drive.
   with directory,
         disk types,
         errors,
         text io;
   use disk types,
         directory;
   procedure display dir is
      current a dir
                         : pathname;
      last valid char
                           : natural;
      current name error : errors.extended errors;
   begin
      current name (
      for drive => a,
      name
                 => current a dir,
      last
                  => last valid char,
      error
                 => current name error
```

text\_io.put\_line (current\_a\_dir (1 .. last\_valid\_char));

end:

## Package directory

Package disk provides operations on disk devices. **SPECIFICATION** with disk types; package disk is procedure reset; function set default (to drive : disk types.drive\_id) return disk\_types.logical\_drive\_count; function get default return disk\_types.drive\_id; -- Get File Allocation Information procedure get\_allocation\_info ( for drive : disk\_types.drive\_designator; sectors : out natural; bytes per\_sector : out natural; clusters : out natural; fat id\_byte : out disk types.identification; : out boolean error **)**; procedure set verification (is on : boolean); function verification is on return boolean; procedure get free space ( for drive : disk\_types.drive\_designator; sectors per cluster : out natural; available clusters : out natural; bytes\_per\_sector : out natural; total clusters : out natural; error : out boolean procedure get free space ( : disk\_types.drive\_designator; for drive free bytes : out long integer; total space : out long integer; : out boolean error

#### 8.1 Procedure Reset

); end disk;

Procedure reset flushes all file buffers to disk without closing any files.

#### procedure reset;

This call corresponds to Interrupt 16#21#, function 16#0D#.

## 8.2 Function Set Default

```
Function set_default sets the current default drive and returns the number of logical drives installed.
```

```
function set default (to drive : disk_types.drive id)
  return disk_types.logical_drive_count;
```

Logical drive means any block device: ram disk, floppy disk, or hard disk.

This call corresponds to Interrupt 16#21#, function 16#0E#.

An example of using package disk with function set default follows.

```
-- set the current drive to the 'A' drive.
--
with disk, disk_types;
use disk, disk_types;
procedure disk_test1 is
   count : disk_types.logical_drive_count;
begin
   count := set_default (to_drive => a);
end;
```

# 8.3 Function Get\_Default

Function get default returns the drive ID of the default disk drive.

```
function get_default return disk_types.drive_id;
```

This call corresponds to Interrupt 16#21#, function 16#19#.

# 8.4 Procedure Get\_Allocation\_Info

Procedure get\_allocation\_info obtains information about the specified disk drive.

This call corresponds to Interrupt 16#21#, function 16#1C#.

# 8.5 Procedure Set Verification

Procedure set\_verification turns on or off the read-after-write verification of all data written to disk.

```
procedure set_verification (is_on : boolean);
```

This call corresponds to Interrupt 16#21#, function 16#2E#.

An example of using package disk with procedure set\_verification follows.

```
-- set verification to on
--
disk.set_verification (is_on => true);
```

# 8.6 Function Verification\_Is\_On

Function verification\_is\_on determines whether the read—after—write verification flag is on or off.

function verification\_is\_on return boolean;

This call corresponds to Interrupt 16#21#, function 16#54#.

# 8.7 Procedure Get\_Free Space

Procedure get\_free\_space obtains selected information for the specified disk drive.

An error may be caused by an invalid drive designator.

This call corresponds to Interrupt 16#21#, function 16#36#.

# 8.8 Procedure Get\_Free\_Space

Procedure **get\_free\_space** obtains the amount of total space and amount of free space for the specified disk drive.

An error may be caused by an invalid drive designator.

This call corresponds to Interrupt 16#21#, function 16#36#.

An example of using package disk with procedure get\_free\_space follows.

```
-- Get and display the used space of the 'A' disk drive.

with disk,
    text_io;
use disk;
procedure display_used_on_a is
```

```
total_free : long_integer;
  disk size : long_integer;
  space_error : boolean;
begin
  get_free_space (
                  => a;
     for drive
                 => total_free;
     free byte
     total_space => disk_size;
     error
                  => space_error
   );
   if not space_error then
     text io.put_line (
        "Total Used is " &
        long_integer'image (disk_size - total_free));
   end if;
end;
```

# Chapter 9 Package Disk\_Types

This package contains type declarations used by the disk and directory operation packages absolute\_disk, directory, and disk.

#### **SPECIFICATION**

```
package disk_types is

type drive_designator is (
    current_drive, a, b, c, d, e,
    f, g, h, i, j, k, l, m, n, o,
    p, q, r, s, t, u, v, w, x, y,
    z
    );

subtype drive_id is drive_designator range a .. z;

subtype logical_drive_count is integer range 0 .. 255;
subtype identification is integer range 0 .. 255;
end disk_types;
```

#### Package disk\_types

# **Chapter 10 Package Equipment**

Package equipment provides a function that enumerates the system—recognized devices and facilities present on the machine.

#### **SPECIFICATION**

```
package equipment is
  type mode is (not_used, color40, color80, mono80);
  type equipment list is
     record
      printers
                         : integer;
      serial_printer
                       : boolean;
      game_adapter
                        : boolean;
      rs232_ports
                        : integer;
                       : boolean;
      dma_present
      diskette drives : integer;
      initial video mode : mode;
      system ram : integer;
      math_coprocessor : boolean;
      diskette present : boolean;
      end record;
   function list return equipment_list;
end equipment;
```

#### 10.1 Function List

Function list returns the list of currently installed equipment.

```
function list return equipment_list;
```

The equipment list information corresponds to the word at address 0000:0410H.

This call corresponds to Interrupt 16#11#.

An example of using package equipment with function list follows.

```
-- determine if a game adapter is present.
with equipment; use equipment;
procedure equipment_test1 is
    my_equipment : equipment_list;
begin
    my_equipment := equipment.list;
    if my_equipment.game_adapter then
        null;
    end if;
end:
```

```
Marriago Seprencial de la casa de la
Professional Andreas (Professional Francisco de la California de California de California de California de Cal
                                                                                                                                                                        SHOOT BUTTON OF FRAME
                                                                                                                                                                             The Bulk of the Control of the Contr
                                               . (Glorow - Baroliso - Marketolybers - the for the such asse
                                                                                                                               Tel dell Austrica i regul
                                                                                                                                                                          13.0000322 1 1
                                                                                                                                                                 FAR BELLEO
                                                                                                   as ollo ac
                                                                                                                                             asslood .
                                                                                                                                                   Longle State
                                                                                                  retui
                                                                                                                                                      PARTY SERVER
                                                                                                    1.44 (b) 65 --
                                                                                                                                                      John Highwa
                                                                                                    Termedice :
                                                                                                                                          BUYET BUYET
                                                                                                         The in a sound and the Esterate
                                                                                                                                                      i mina ocumbaire.
                                                                                                   . eggs: :
                                                                                                  trible of
                                                                                                                                      CLER PRODUCT PROCESSES
                                                                                                  Carried to
                                                                                                                                        Carrenal station
                                                                                                                                                             $20 KUNDU - 10 1986
                                                                                        diel demograps repost dell dell'ent
                                                                                                                                                              i i dan masu tejuk bak
                                                                                                                                                                TELL HOLLOW THE SALE
                                                                              LANCE OF EAST POLICE AND LOS OF CORRESTA MEMBERS CONTINUED
                                                                                                 parti inagrape marks come mark
                                                    and the EPO rest loss is into year or charge ance in Sale record and the constant of
                                                                                                                                  AND ENGINEERS SEEN THE CASE OF
                                                                  o realist crist interestablish dream depth that do go other from some
                                                                                 tonemicope who constituted and
                                                                                                                         CO CARACT TATE CALL S
                                                                                                           tack companies to the second
                                                                                                      Constanting to the second of the second
                                                                                                 amid the igsec large deport lives in a
```

1.3737

# **Chapter 11 Package Errors**

Package errors provides PC-DOS status declarations and for version 3.0 or later of PC-DOS, a procedure to get extended status information.

#### **SPECIFICATION**

```
package errors is
   type extended_errors is (
       invalid function,
       file_not_found,
       path not found,
       no handle available,
       access denied,
       invalid handle,
       memory_blocks_destroyed,
       insufficient_memory,
       invalid_memory_block,
       invalid_environment,
       invalid format,
       invalid file access,
       invalid data,
       reserved14,
       invalid drive,
       remove_current_directory,
      not same device,
      no_more_files,
      disk write protected,
      unknown unit,
      drive_not_ready,
      command not defined,
      disk_data_error,
      bad_structure_length,
      seek_error,
      unknown media type,
      sector_not_found,
      printer_outof_paper,
      write_error,
      read_error,
      general_failure,
      file_sharing_violation,
      file_locking_violation,
      invalid disk change,
      no_fcb_available,
```

```
file already exists,
     reserved81,
     cannot make,
     interrupt failure,
     unknown error
   );
 type class_code is (
                           temporary_situation,
     out_of_resource,
                           internal_dos_error,
     authorization,
                           system_sw_error,
     hardware failure,
                            not found,
     application_error,
                            item locked,
     bad format,
                            already exists,
     media error,
     unknown
 type action_code is (
                            try again later,
     try again,
                            shut_down_program,
      user to fix,
                            ignore_error,
      shut_down_immediate,
                            unknown
      retry_after_fix,
  type locus_code is (
                            block device,
      unknown,
                            serial_device,
      network related,
      memory_related
    );
  -- DOS 3.0 procedure
  procedure get_extended_info (
                : out extended_errors;
      error
                 : out class_code;
      class
                 : out action_code;
      action
                 : out locus_code
      locus
     );
end errors;
```

# 11.1 Procedure Get\_Extended\_Info

Procedure get\_extended\_info obtains detailed information after a previously unsuccessful system call. The information may suggest action that the program should take.

```
procedure get_extended_info (
    error : out extended_errors;
    class : out class_code;
    action : out action_code;
    locus : out locus_code
```

This function requires DOS 3.0 or higher. Execution of this function with a lower revision causes the exception revision.incorrect\_dos\_version to be raised.

This call corresponds to Interrupt 16#21#, function 16#59#. An example follows. with errors, directory; use errors; procedure errors\_test1 is make\_error : extended\_errors; error : extended\_errors; class : class\_code; action : action\_code; locus : locus\_code; begin make\_error := directory.make (name => "A:\Junk"); if make\_error /= ok then get\_extended\_info (error, class, action, locus); end if; end;

#### Package errors

Package **file\_io** provides file input—output, file attribute manipulation, and wildcard search operations. Note that most of the essential input—output facilities provided in this package are available in a largely system—independent standard package, **text\_io**, which is distributed with the Meridian Ada compiler.

Be aware that the Ada predefined type character supports only seven—bit ASCII characters. If it is necessary to manipulate all eight bits of a byte, then pragma suppress should be used when using objects of type character (to avoid the exception constraint\_error), or a different type altogether (e.g. common\_display\_types.extended\_ascii) should be used.

#### **SPECIFICATION**

```
with errors,
     time,
     system;
package file_io is
   type file_handle is new integer;
   -- Standard I/O Handles
   stdin : constant file handle := 0;
                                         -- CON
   stdout : constant file handle := 1;
                                         -- CON
   stderr : constant file handle := 2;
                                         -- CON
   stdaux : constant file handle := 3;
                                         -- AUX
   stdlist : constant file handle := 4;
   type file_attribute_types is (
       read_only,
                     hidden,
                                   system,
      volume_label, subdirectory, archive
  type file_attributes is array (file_attribute_types)
      of boolean:
   - Access Code Values
  type access mode is (
      read only,
      write_only,
      read_write
    );
  type sharing mode is (
      compatibility, deny_read_write, deny_write,
      deny_read,
                     deny_none
    );
```

```
type access_code is
   record
    inheritance_flag : boolean
                                   := true;
    -- DOS 3.0 only
                     : sharing_mode := compatibility;
    sharing
    -- DOS 3.0 only
                    : boolean := false;
    reserved bit
    -- DOS 3.0 only
    file_access_mode : access_mode := read_write;
   end record;
type standard_io is (input, output);
type method_code is (
    beginning,
    current location,
    current_end
   );
 subtype timestamp_data is time.packet;
 type transfer_data is limited private;
 type file data is
    record
              : string (1..12);
     attributes : file_attributes;
     timestamp : timestamp_data;
     file_size : long_integer; -- in bytes
    end record;
 procedure create (
             : string;
     name
               : out file_handle;
     handle
               : out errors.extended errors;
     attributes : file_attributes :=
                    (others => false)
    );
  -- DOS 3.0
  procedure create_temporary (
      pathname : string;
               : out file_handle;
      handle
      temp file : out string;
                : out natural;
      last
      error : out errors.extended_errors;
      attributes : file_attributes :=
                     (others => false)
    );
```

```
-- DOS 3.0
 procedure create_temporary (
     handle
            : out file handle;
     temp_file : out string;
    last
              : out natural;
           : out errors.extended_errors;
     error
    attributes : file_attributes :=
                    (others => false)
  );
 -- DOS 3.0
procedure create_new (
    name
             : string:
    handle
              : out file handle;
    error : out errors.extended_errors;
    attributes : file_attributes :=
                   (others => false)
  );
procedure open (
    name : string;
    handle : out file handle;
    error : out errors.extended errors;
    code : access_code := (
        inheritance_flag => true,
        sharing
                        => compatibility,
        reserved bit
                         => false,
        file_access_mode => read_write
      )
  );
function close (handle : file handle)
    return errors.extended_errors;
function cooked (for_standard : standard io)
    return errors.extended errors;
function raw
                (for standard : standard io)
    return errors.extended_errors;
procedure read (
    handle
                  : file handle;
    bytes to read : natural;
    buffer address : system.address;
    bytes_read : out natural;
    error
                  : out errors.extended_errors
  );
procedure write (
   handle
                   : file handle;
   bytes_to_write : natural;
   buffer address : system.address;
   bytes_written : out natural;
   error
                 : out errors.extended_errors
 );
```

```
function delete (name : string) return
    errors.extended_errors;
function rename (
    old name : string;
    new_name : string
  ) return errors.extended_errors;
procedure move_file_pointer (
              : file handle;
    handle
    offset method : method code;
    offset_value : long_integer;
    new_pointer : out long_integer;
               : out errors.extended_errors
    error
  );
 -- Get and Set File Attributes
procedure get_attributes (
     name : string;
     attributes : out file_attributes;
     error : out errors.extended_errors
   );
 function set_attributes (
     name : string;
     attributes : file_attributes
   ) return errors.extended_errors;
  -- Get and Set File Date and Time
 function set_file_time (
     handle : file handle;
     time : timestamp data
   ) return errors.extended_errors;
 procedure get_file_time (
     handle : file handle;
      time : out timestamp_data;
     error : out errors.extended_errors
    );
  procedure dup_filehandle (
      handle : file_handle;
      new handle : out file handle;
      error : out errors.extended_errors
  procedure cdup_filehandle (
      handle1 : file_handle;
handle2 : file_handle;
error : out errors.extended_errors
    );
```

```
- File Search functions
       procedure find first (
           name_template : string;
           transfer_area
                              : out transfer data;
           file info
                              : out file data;
           error
                              : out errors.extended_errors;
           search_attributes: file_attributes :=
                                   (others => false)
         );
      procedure find_next (
           transfer_area : in out transfer_data;
           file info : out file data;
           error
                           : out errors.extended errors
         ):
   private
      type transfer_data is array (0 .. 21) of integer;
   end file io;
12.1 Procedure Create
Procedure creates a file in the current or specified directory, and opens it for program use.
   procedure create (
       name
                    : string;
                    : out file handle;
                   : out errors.extended_errors;
       attributes : file_attributes :=
                         (others => false)
     );
If the file already exists, then it is truncated to zero. The file is opened for read/write access.
File attributes volume and subdirectory are ignored by this call.
This call corresponds to Interrupt 16#21#, function 16#3C#.
Possible errors are:

    errors.path not found

      errors.no_handle_available
      errors.access_denied
An example of using package file_io with procedure create follows.
   -- create "Test.Out" with normal attributes in the
   -- current directory on the default drive.
  with file_io, errors, text_io;
  use file io, errors;
```

procedure fio test1 is

# 12.2 Procedure Create\_Temporary

There are two overloaded versions of procedure **create\_temporary** disambiguated by the presence or absence of the parameter *pathname*. The first version creates a temporary file in the directory specified by the *pathname*; the second version creates a temporary file in the current directory.

### 12.2.1 Using Pathname

With a pathname parameter, procedure create\_temporary creates a temporary file in the directory specified by pathname, and opens the file.

The parameter temp\_file is set to the path and name of the temporary file. Parameter last is the index position of the last valid character of the temp\_file string.

The temporary file is not automatically deleted at program completion.

The file is opened for read/write access.

File attributes volume and subdirectory are ignored by this call.

This function requires DOS 3.0 or higher. Execution of this function with a lower revision causes the exception revision.incorrect\_dos\_version to be raised.

This call corresponds to Interrupt 16#21#, function 16#5A#.

Possible errors are:

- errors.path\_not\_found
- errors.access\_denied

```
An example of using package file_io with procedure create_temporary follows.
  -- Create a temporary file in the directory specified
  -- by "A:\TEMP.DIR" (a:\temp.dir must already exist)
  with file_io, errors;
  use file io;
  procedure fio test2 is
    temp_handle : file_handle;
    temp_filename : string;
    last_char : natural;
    create_error : errors.extended_errors;
  begin
    create_temporary (
        pathname => "A:\TEMP.DIR",
                 => temp_handle,
        handle
        temp_file => temp_filename,
        last
                 => last char,
        error
                 => create error
      );
  end:
```

### 12.2.2 Using the Current Directory

Without a pathname parameter, procedure create temporary creates a temporary file in the current directory on the default disk drive and opens the file.

The parameter *temp\_file* is set to the name of the temporary file. Parameter *last* is the index position of the last valid character of the *temp\_file* string.

The temporary file is not automatically deleted at program completion.

The file is opened for read/write access.

File attributes volume and subdirectory are ignored by this call.

This function requires DOS 3.0 or higher. Execution of this function with a lower revision causes the exception revision.incorrect\_dos\_version to be raised.

This call corresponds to Interrupt 16#21#, function 16#5A#.

Possible errors are:

- errors.path\_not\_found
- errors.access\_denied

# 12.3 Procedure Create\_New

Procedure create\_new creates a file in the current or specified directory, and opens the file. If the file already exists, then the file\_already\_exists error is returned.

The file is opened for read/write access.

File attributes volume and subdirectory are ignored by this call.

This call corresponds to Interrupt 16#21#, function 16#5B#.

This function requires DOS 3.0 or higher. Execution of this function with a lower revision causes the exception revision.incorrect\_dos\_version to be raised.

Possible errors are:

```
errors.path_not_found
```

- errors.no\_handle\_available
- errors.access\_denied
- errors.file\_already\_exists

An example of using package file\_io with procedure create\_new follows.

```
-- Create "TEST" as a Read_Only file in the current
-- directory.
with file_io, errors,
use file io;
procedure fio_test3 is
   new_handle : file_handle;
   create error : errors.extended errors;
   file_attr : file_attributes :=
                    (others => false);
begin
   file attr (read_only) := true;
   create new (
                 => "Test",
       name
                 => new handle,
       handle
       error => create_error,
       attributes => file_attr
     );
 end:
```

# 12.4 Procedure Open

Procedure open opens an existing file.

```
procedure open (
   name : string;
handle : out file_handle;
error : out errors.extended_errors;
code : access_code := (
   inheritance_flag => true,
        sharing => compatibility,
        reserved_bit => false,
        file_access_mode => read_write
   )
);
```

For DOS revision 2, the *file\_access\_mode* is the only parameter actually used; the parameters *inheritance\_flag* and *sharing* are ignored. All the parameters are used in revision 3.0 and higher.

Normal files and files with the system and hidden attributes can be opened with this call.

This call corresponds to Interrupt 16#21#, function 16#3D#.

Possible errors are:

- errors.invalid function
- errors.file\_not\_found
- errors.path\_not\_found
- errors.no\_handle\_available
- errors.access denied
- errors.invalid\_file access

An example of using package file\_io with procedure open follows.

```
-- Open the file "TEST.OUT" in the current directory with
-- the default settings.

with file_io, errors;
use file_io;
procedure fio_test4 is

test_handle : file_handle;
    open_error : errors.extended_errors;
begin
    open ("Test.Out", test_handle, open_error);
end;
```

#### 12.5 Function Close

Function close closes a file whose handle was previously returned by a successful call to open, create, create\_temporary, or create\_new.

```
function close (handle : file_handle)
  return errors.extended_errors;
```

Closing flushes the file's internal buffers to disk and releases the handle for reuse.

The file's date and time is updated if the file was modified.

This call corresponds to Interrupt 16#21#, function 16#3E#.

A possible error follows.

· errors.invalid handle

An example follows.

```
-- Close the file associated with the specified handle
-- and notify user if in error.

with file_io, errors;
use file_io, errors;
procedure fio_test5 (test_handle : file_handle) is

begin
   if close (handle => test_handle) /= ok then
        text_io.put_line ("Error on Close");
   end if;
end;
```

#### 12.6 Function Cooked

Function cooked sets the cooked bit in the device driver information word for the specified standard device.

```
function cooked (for_standard : standard_io)
    return errors.extended_errors;
```

Characters that require special action are checked on input or output.

Character devices perform their input and output in the cooked mode by default.

This call corresponds to Interrupt 16#21#, function 16#44#.

# 12.7 Function Raw

Function raw sets the raw bit in the device driver information word for the specified standard device.

```
function raw (for_standard : standard_io)
    return errors.extended_errors;
```

Checking for Control—C (^C) or any other control characters is turned off. This increases display speed. The system does not take any action on special characters in the input stream.

This call corresponds to Interrupt 16#21#, function 16#44#.

# 12.8 Procedure Read

Procedure read reads from a file or device associated with the specified handle.

```
procedure read (
    handle : file_handle;
    bytes_to_read : natural;
    buffer_address : system.address;
    bytes_read : out natural;
    error : out errors.extended_errors
);
```

The number of bytes actually read is returned in bytes\_read. This value may be less than the value specified in bytes\_to\_read if:

- an error occurs.
- the end of file is encountered.
- an end-of-line sequence (carriage-return/line-feed) is read from a character device that is in cooked mode. The CR/LF pair is read into the buffer and is counted.

The read buffer associated with the buffer\_address must be large enough for the number of bytes specified for reading.

The file position pointer is updated.

This call corresponds to Interrupt 16#21#, function 16#3F#.

Possible errors are:

- errors.access\_denied
- errors.invalid handle

An example of using package file\_io with procedure read follows.

```
-- Read a maximum of 80 bytes or until a carriage-return
-- is detected from the standard input device.
with file_io, errors;
procedure fio test6 is
   buffer
               : string (1 .. 80);
   qty_read
              : natural;
   read_error : errors.extended_errors;
begin
   file io.read (
      handle
                     => stdin,
      bytes_to_read => 80,
      buffer_address => buffer (1)'address,
      bytes_read => qty_read,
       error
                     => read error
    );
end:
```

#### 12.9 Procedure Write

Procedure write writes to a file or device associated with the specified handle.

```
procedure write (
    handle : file_handle;
    bytes_to_write : natural;
    buffer_address : system.address;
    bytes_written : out natural;
    error : out errors.extended_errors
);
```

The number of bytes actually written is returned in bytes\_written. This value may be less than the number specified in bytes\_to\_write if an error occurs (e.g. the disk is full).

The file position pointer is updated.

This call corresponds to Interrupt 16#21#, function 16#40#.

Possible errors are:

- errors.access\_denied
- errors.invalid handle

An example of using package file\_io with procedure write follows.

```
-- Write the test string to a previously-opened file.
with file io, errors;
use file_io;
procedure fio_test7 (open_handle : file_handle) is
   qty_written : natural;
   buffer : string (1 .. 14) := "This is a Test";
   write_error : errors.extended_errors;
begin
   write (
                     => open handle,
       handle
       bytes_to_write => buffer'length,
       buffer_address => buffer (1)'address,
       bytes_written => qty_written,
                     => write_error
       error
     );
 end;
```

## 12.10 Function Delete

Function delete deletes a file.

```
function delete (name : string) return errors.extended_errors;
```

Wildcard characters are not valid.

Files that have the *read\_only* attribute set can not be deleted until cleared by a call to *set\_attributes*. This call corresponds to Interrupt 16#21#, function 16#41#.

Possible errors are:

- errors.file\_not\_found
- errors.access\_denied

#### 12.11 Function Rename

Function rename renames a file or moves the directory entry of a file to another directory.

```
function rename (
   old_name : string;
   new_name : string
) return errors.extended_errors;
```

Wildcard characters are not valid.

This call corresponds to Interrupt 16#21#, function 16#56#. Possible errors are:

```
errors.file_not_found
        errors.path not found
        errors.access_denied
        errors.not_same device
An example of using package file_io with function rename follows.
  -- Move "file.dat" to directory "test2" and rename
   -- it to "old.dat".
  with file io, errors;
  use file io;
  procedure fio_test8 is
    rename_error : errors.extended_errors;
  begin
    rename error :=
      rename (
         old_name => "C:\TEST.DIR\FILE.DAT",
         new_name => "C:\TEST2.DIR\OLD.DAT"
```

# 12.12 Procedure Move\_File\_Pointer

Procedure move\_file\_pointer sets the file position for the next input—output operation. The position may be set to the start of the file, to the end of the file, or to a location relative to the current file position.

```
procedure move_file_pointer (
    handle : file_handle;
    offset_method : method_code;
    offset_value : long_integer;
    new_pointer : out long_integer;
    error : out errors.extended_errors
);
```

The value returned in new\_pointer is the byte offset from the beginning of the file.

It is possible to set the file pointer to a location before the start of the file. If this is done, any attempt to read or write to the file causes an error. If the file pointer is positioned beyond the end of the file, a subsequent write to the file inserts "padding" (garbage characters) between the previous end of file and the selected file position.

This call corresponds to Interrupt 16#21#, function 16#42#.

Possible errors are:

end;

- errors.invalid\_function
- · errors.invalid handle

An example of using package file\_io with procedure move\_file\_pointer follows.

```
-- Move the file pointer to the end of the file,
 -- and display that position.
 with file_io, errors, text_io;
 use file_io;
 procedure fio_test9 (open_handle : in file_handle) is
    current_end : method_code;
    file size : long integer;
    move_error : errors.extended_errors;
    move file_pointer (
                 => open_handle,
         handle
        offset_method => current_end,
         offset value => 0,
        new_pointer => file_size,
         error => move_error
    text_io.put_line (long_integer'image (file_size));
  end;
12.13 Procedure Get_Attributes
Procedure get_attributes obtains the attributes of a file in the current or specified directory.
   procedure get_attributes (
                     : string;
       name
       attributes : out file_attributes;
error : out errors.extended_errors
     );
This call corresponds to Interrupt 16#21#, function 16#43#.
Possible errors are:
      errors.file_not_found
       errors.path_not_found
An example of using package file_io with procedure get_attributes follows.
   -- Determine if the Read_Attribute is set for the file "TEST.DAT".
    with file io, errors;
    use file io, errors;
    procedure fio_test10 is
```

```
attr : file_io.file_attributes;
         error : errors.extended_errors;
         get_attributes ("TEST.DAT", attr, error);
         if error = ok and then attr (read_only) then
            -- user defined.
            null;
         end if;
     end:
  12.14 Function Set_Attributes
 Function set_attributes sets the attributes on a file in the current of specified directory.
     function set_attributes (
         name
                      : string;
         attributes : file_attributes
       ) return errors.extended_errors;
 The volume_label and subdirectory attributes cannot be set.
 This call corresponds to Interrupt 16#21#, function 16#43#.
 Possible errors are:

    errors.file_not_found

         errors.path_not_found
       errors.access_denied
 12.15 Function Set_File_Time
 Function set_file_time modifies the time and date of a file associated with a specified handle.
    function set file time (
        handle : file handle;
                : timestamp data
      ) return errors.extended_errors;
This call corresponds to Interrupt 16#21#, function 16#57#.
A possible error follows.
          errors.invalid_handle
12.16 Procedure Get_File_Time
Procedure get_file_time obtains the time and date of a file associated with a specified handle.
   procedure get_file_time (
        handle : file handle;
       time : out timestamp_data;
       error : out errors.extended_errors
This call corresponds to Interrupt 16#21#, function 16#57#.
```

A possible error follows.

errors.invalid\_handle

# 12.17 Procedure Dup\_FileHandle

Procedure dup\_filehandle duplicates a currently open handle.

```
procedure dup_filehandle (
   handle : file_handle;
   new_handle : out file_handle;
   error : out errors.extended_errors
);
```

The new\_handle refers to the same file or device at the same position as the original handle. If the file position pointer is moved by one handle, the other handle is also moved. This call corresponds to Interrupt 16#21#, function 16#45#.

Possible errors are:

- errors.no\_handle\_available
- errors.invalid\_handle

# 12.18 Procedure Cdup\_FileHandle

Procedure cdup\_filehandle duplicates a file handle.

```
procedure cdup_filehandle (
    handle1 : file_handle;
    handle2 : file_handle;
    error : out errors.extended_errors
);
```

Handle2 is made to refer to the same file or device as handle1.

A file previously associated with handle2 is closed if the file was open.

If the file position pointer is moved by one handle, the other handle is also moved.

This call corresponds to Interrupt 16#21#, function 16#46#.

Possible errors are:

- errors.no\_handle\_available
- errors.invalid\_handle

# 12.19 Procedure Find\_First

Procedure **find\_first** finds the first occurrence of *name\_template* with the specified attributes in the current or specified directory.

Name\_template may contain wildcard characters.

If search\_attributes contains volume\_label, then only volume\_label files are found. Other files are found if search\_attributes specifies attribute combinations not including volume\_label.

This call corresponds to Interrupt 16#21#, function 16#4E#.

Possible errors are:

- errors.path\_not\_found
- errors.no\_files

For an example see procedure find\_next.

# 12.20 Procedure Find\_Next

Procedure find next finds the next occurrence of the name\_template specified in the call to find first.

```
procedure find_next (
    transfer_area : in out transfer_data;
    file_info : out file_data;
    error : out errors.extended_errors
);
```

Transfer\_area must contain the information set there by a previous call to find\_first or find\_next.

A possible error follows.

• errors.no\_more\_files

An example of using package file\_io with procedure find\_next follows.

```
-- General routine to display files on disk.
with file_io, errors, text_io;
use file_io, errors;
procedure display_files (template : string) is
  search data : transfer data;
  file_information : file_data;
                 : errors.extended_errors;
  find_error
begin
  find first (
      name_template => template,
      transfer_area => search_data,
      file info => file information,
                    => find error
      error
    );
  if find error = ok then
    text_io.put_line (file_information.name);
    loop
      find next (
           transfer_area => search_data,
           file_info => file_information,
                        => find error
           error
        );
```

```
if find_error = ok then
    text_io.put_line (file_information.name);
else
    exit;
end if;
end loop;
end if;
end;
```

Package interrupt allows calls to the interrupt vectors. Refer to your technical reference manuals for complete details of these interrupts and the required parameters.

#### **SPECIFICATION**

```
package interrupt is
   type interrupt_range is
     new integer range 0 .. 16#ff#;
   type registers is
      record
             : integer;
              : integer;
             : integer;
             : integer;
       dж
             : integer;
       ds
             : integer;
       es
             : integer;
        si
             : integer;
       carry : integer; -- 0 or 1
        flags : integer;
       end record;
   procedure vector (
                        : interrupt_range;
        register_block : in out registers
      );
 end interrupt;
```

#### 13.1 Procedure Vector

Procedure **vector** takes an interrupt number and a list of register values. It performs the specified interrupt and returns the states of the registers following the interrupt. The CS, SS, SP, and BP registers may not be changed through this call.

```
procedure vector (
    on : interrupt_range;
    register_block : in out registers
);
```

#### Package interrupt

Only those registers actually used by the interrupt need to be set. The flags component of the registers record is arranged as:

Bit	Description	Bit	Description
0	CF ·	8	TF
1	undefined	9	IF
2	PF	Α	DF
3	undefined	В	OF
4	AF	C,D	IOPL*
5	undefined	E	NT*
6	ZF	F	undefined
7	SF		

<sup>\*80286</sup> flag

Note that the carry flag is provided separately, since it is the flag most often used to indicate return status in DOS calls.

Warning: Incorrect use of any interrupt may have unpredictable or disasterous results.

An example of using package interrupt with procedure vector follows.

This example shows how to use interrupt.vector to call a DOS function. Note that this particular DOS call is also available as tty.put.

Package memory provides functions to allocate and deallocate memory. Functions are also present to manipulate allocated memory areas and to obtain information about available memory resources.

Note that paragraphs are 16-byte quantities.

#### **SPECIFICATION**

```
with system,
     errors;
package memory is
   type memory_segment is new integer;
   type segment_offset is new integer;
   function make (
       segment : memory_segment;
       offset : segment_offset
     ) return system.address;
   procedure split (
       dos_address : system.address;
                  : out memory segment;
       segment
                    : out segment_offset
       offset
     );
   procedure allocate (
       paragraphs_requested : natural;
                             : out memory_segment;
       segment
       largest_block_avail : out natural;
                             : out errors.extended_errors
       error
     );
    function release (segment : memory_segment)
        return errors.extended errors;
   procedure modify (
        paragraphs_requested : natural;
                             : memory_segment;
        segment
        largest block avail : out natural;
                              : out errors.extended_errors
        error
      );
    function installed return integer; -- in K-bytes
    function used return natural; -- in Paragraphs
 end memory;
```

#### 14.1 Function Make

Function make returns an address from the specified segment and offset.

```
function make (
    segment : memory_segment;
    offset : segment_offset
) return system.address;
```

This does not correspond to a particular DOS function.

### 14.2 Procedure Split

Procedure split obtains the segment and offset from a dos\_address.

```
procedure split (
   dos_address : system.address;
   segment : out memory_segment;
   offset : out segment_offset
);
```

This does not correspond to a particular DOS function.

#### 14.3 Procedure Allocate

Procedure allocate dynamically allocates memory of the number of paragraphs\_requested and returns the segment paragraph of the allocated memory block.

```
procedure allocate (
    paragraphs_requested : natural;
    segment : out memory_segment;
    largest_block_avail : out natural;
    error : out errors.extended_errors
);
```

If the procedure fails to allocate the amount of memory requested, then largest\_block\_avail contains the size of the largest available memory block.

A paragraph is 16 bytes in size.

This call corresponds to Interrupt 16#21#, function 16#48#.

Possible errors are:

- errors.memory\_blocks\_destroyed
- errors.insufficient\_memory

An example of using package memory with procedure allocate follows.

```
-- Allocate a block of 64k
--
with memory, errors, system, tty;
use memory, errors;
procedure allocate_64k is
```

```
block_segment : memory_segment;
 biggest_block : natural;
 allocate_error : errors.extended_errors;
 starting_address : system.address;
begin
  allocate (
      paragraphs_requested => 4096,
                           => block segment,
      segment
      largest_block_avail => biggest_block,
                           => allocate error
    );
  if allocate error = ok then
     starting address :=
       make (
       segment => block_segment,
       offset => 0
       );
  else
     tty.put ("Error is : " &
       errors.extended_errors'image (allocate_error));
  end if;
end;
```

#### 14.4 Function Release

Function release releases a memory block that was previously allocated by allocate.

```
function release (segment : memory_segment)
   return errors.extended_errors;
```

The segment to release must be the same as the segment obtained from allocate. An attempt to release an invalid segment has unpredictable results.

This call corresponds to Interrupt 16#21#, function 16#49#.

Possible errors are:

- errors.memory\_blocks\_destroyed
- errors.invalid\_memory\_block

# 14.5 Procedure Modify

Procedure **modify** dynamically increases or decreases the size of a memory block that was previously allocated by **allocate**.

```
procedure modify (
    paragraphs_requested : natural;
    segment : memory_segment;
    largest_block_avail : out natural;
    error : out errors.extended_errors
);
```

If the procedure fails to modify the size of the specified segment, then largest\_block\_avail contains the size of the largest available block.

#### Package memory

This call corresponds to Interrupt 16#21#, function 16#4A#.

Possible errors are:

- errors.memory\_blocks\_destroyed
- errors.insufficient memory
- errors.invalid\_memory\_block

#### 14.6 Function Installed

Function installed returns the number of kilobytes of memory currently installed.

```
function installed return integer; -- in K-bytes
```

This call corresponds to Interrupt 16#12#.

An example of using package memory with function installed follows.

```
-- get the total installed memory.
--
with memory, tty;
procedure show_total is
begin
    tty.put ("Total Memory is : " &
        integer'image (memory.installed));
end;
```

#### 14.7 Function Used

Function used returns the size of the program at startup, excluding memory allocated during the program's execution.

```
function used return natural; -- in Paragraphs
```

The value returned is in terms of paragraphs.

The value returned can be used with program\_control.resident\_quit.

Information about the amount of memory used is obtained from the Program Segment Prefix; there is no corresponding system or BIOS call.

#### **SPECIFICATION**

```
package port is
  function in_word (port_number : integer)
    return integer;
  function in_byte (port_number : integer)
    return integer;
  procedure out_word (
        port_number : integer;
        data : integer
    );
  procedure out_byte (
        port_number : integer;
        data : integer
    );
end port;
```

#### **USAGE**

Package port allows byte or word input and output to the specified port.

Since the actual devices corresponding to particular 80x86 I/O ports may vary from machine to machine, you should consult the hardware reference materials or BIOS listing for your system in order to find out, for example, which I/O port corresponds to a serial chip or a speaker.

The document describing the hardware and BIOS for the IBM PC/AT is: Personal Computer AT Technical Reference, IBM document number 6280070.

# 15.1 Function In Word

Function in word transfers a word from the specified input port.

```
function in_word (port_number : integer)
  return integer;
```

This does not correspond to a particular DOS function; it uses the 80x86 IN instruction.

## 15.2 Function In Byte

Function in byte transfers a byte from the specified input port.

```
function in_byte (port_number : integer)
  return integer;
```

This does not correspond to a particular DOS function; it uses the 80x86 IN instruction.

# 15.3 Procedure Out\_Word

procedure out\_word transfers a word to the specified output port.

```
procedure out_word (
    port_number : integer;
    data : integer
);
```

This does not correspond to a particular DOS function; it uses the 80x86 OUT instruction.

# 15.4 Procedure Out\_Byte

Procedure out\_byte transfers a byte to the specified output port.

```
procedure out_byte (
    port_number : integer;
    data : integer
);
```

This does not correspond to a particular DOS function; it uses the 80x86 OUT instruction.

Package program control provides functions that exert control over the executing program, that execute other programs, or that obtain information about resident programs.

#### **SPECIFICATION**

```
with errors,
    memory;
package program_control is
   subtype byte is integer range 0 .. 255;
   type program end is (
       voluntary_end, ctrl_break_end,
       device_error, stay_resident_end
     );
   type state_type is (off, on);
   procedure quit (return_code : byte);
   function resident quit (
                           : byte;
       return code
       reserve paragraphs : natural
     ) return errors.extended_errors;
   procedure get_return_code (
       return code : out byte;
                     : out program_end
       how ended
     );
   function break_status return_state_type;
   procedure set_break_status (to_state : state_type);
   function segment_prefix return memory.memory_segment;
   function execute (
       program name
                           : string;
        command arguments : string
      ) return errors.extended errors;
    function msdos (command line : string)
        return errors.extended errors;
    procedure get_environment_variable(
        variable name : string;
        value
                     : out string;
                      : out natural
        last
      );
 end program_control;
```

#### 16.1 Procedure Quit

Procedure quit terminates the calling program and returns control to PC-DOS or the parent program, yielding a return code.

```
procedure quit (return_code : byte);
```

The return code can be examined with the batch command errorlevel or by the parent program using get\_return\_code.

This call corresponds to Interrupt 16#21#, function 16#4C#.

## 16.2 Function Resident Quit

Function **resident\_quit** terminates the calling program without releasing its memory. Control returns back to PC- DOS or the parent program, yielding a return code.

```
function resident_quit (
    return_code : byte;
    reserve_paragraphs : natural
) return errors.extended_errors;
```

Reserve\_paragraphs is the number of 16 byte units that are to remain resident. An appropriate value to use for reserve\_paragraphs may be obtained by using the function memory.used.

This call corresponds to Interrupt 16#21#, function 16#31#.

## 16.3 Procedure Get Return Code

Procedure **get\_return\_code** obtains the return code of a child program after successful completion of the **execute** procedure.

```
procedure get_return_code (
    return_code : out byte;
    how_ended : out program_end
);
```

The return code can be obtained only once using this call.

This call corresponds to Interrupt 16#21#, function 16#4D#.

## 16.4 Function Break Status

Function break\_status determines the current state of the operating system's Ctrl-Break checking flag.

```
function break_status return state_type;
```

This call corresponds to Interrupt 16#21#, function 16#33#.

## 16.5 Procedure Set\_Break\_Status

Procedure set\_break\_status sets the current state of the operating system's Ctrl-Break checking flag.

```
procedure set_break_status (to_state : state_type);
```

This call corresponds to Interrupt 16#21#, function 16#33#.

# 16.6 Function Segment\_Prefix

Function **segment\_prefix** returns the memory segment of the calling program's program segment prefix (PSP).

```
function segment_prefix return memory.memory_segment;
```

This does not correspond to a particular DOS function.

#### 16.7 Function Execute

Function execute executes the program with the specified program\_name, passing the specified command\_arguments to the program. The calling program is suspended during execution of the child program.

```
function execute (
    program_name : string;
    command_arguments : string
) return errors.extended_errors;
```

Upon completion of the child program, control returns back to the calling program. It is then possible to determine how the child ended by calling the **get\_return\_code** procedure.

This call corresponds to Interrupt 16#21#, function 16#4B#.

Possible errors are:

- errors.file\_not\_found
- errors.access\_denied
- errors.insufficient\_memory
- errors.invalid\_set\_strings
- errors.invalid\_format

#### 16.8 Function Msdos

Function msdos executes the PC-DOS command line interpreter with the specified command line.

```
function msdos (command_line : string)
  return errors.extended_errors;
```

If the specified command\_line is an empty string ("") then a new interactive command line interpreter is invoked. To return to the calling program, type the command exit.

This call corresponds to Interrupt 16#21#, function 16#4B#.

Possible errors are:

- errors.file not\_found
- errors.access\_denied
- errors.insufficient memory
- errors.invalid\_set\_strings
- errors.invalid\_format

An example of using package program\_control with function msdos follows.

```
-- Redirect directory listing to a file.

with program_control,
    errors,
    tty;
use program_control;
procedure list_exe is
```

# 16.9 Procedure Get\_Environment\_Variable

Procedure **get\_environment\_variable** obtains the value associated with a particular environment variable.

The value of the environment variable, if found, is returned in value. The procedure raises the exception constraint\_error if value is too small to hold the value of the environment variable.

Last is set to the index position of the last character in value. Last is set to zero if the specified variable\_name is not in the environment.

# **Chapter 17 Package Revision**

Package revision provides a facility for determining the revision (version) number of PC-DOS.

#### SPECIFICATION

```
package revision is
  incorrect_dos_version : exception;
  type number is
    record
    major : natural;
    minor : natural;
    end record;
  function dos return number;
  function dos return natural; --- major version only
end revision;
```

# 17.1 Exception Incorrect\_Dos\_Version

Exception incorrect\_dos\_version is raised when a system function is invoked that is available only on a later release of PC-DOS than is present on the system.

### 17.2 Function Dos

The overloaded dos functions are disambiguated by the return type. Either a complete revision number may be obtained in the composite type number or just the major revision number may be obtained.

# 17.2.1 Returning Complete Revision Number

The first version of function dos returns the complete version number of the host operating system.

```
function dos return number;
```

The version number is obtained during package elaboration through interrupt 16#21#, function 16#30#. If the antiquated version 1 PC-DOS environment is detected then the exception incorrect\_dos\_ver-sion is raised.

# 17.2.2 Returning Major Revision Only

The second version of function dos returns the major version number of PC-DOS.

```
function dos return natural; -- Major Version Only
```

The version number is obtained during package elaboration through interrupt 16#21#, function 16#30#. If the antiquated version 1 PC-DOS environment is detected then the exception incorrect\_dos\_ver-sion is raised.

An example of using package revision with function dos follows.

```
-- determine if DOS version is lower than 3.0
--
with text_io;
procedure check_revision is
begin
```

### Package revision

```
if revision.dos < 3 then
    text_io.put_line ("Raising Version Exception");
    raise revision.incorrect_dos_version;
end if;
end;</pre>
```

Package time provides procedures to get and set the current system date and time.

SPECIFICATION

```
package time is
                           is natural range 0 .. 23;
   subtype hours range
   subtype minutes range is natural range 0 .. 59;
   subtype seconds range is natural range 0 .. 59;
   subtype hundredths_range is natural range 0 .. 99;
   subtype months_range is positive range 1
                                               .. 12;
   subtype days range is positive range 1
   subtype years range is positive range 1980 .. 2099;
   type day of week is (
               monday, tuesday, wednesday,
       sunday,
       thursday, friday, saturday
     );
   type months is (
                  february, march,
                                      april,
       january,
                  june,
                                      august,
                            july,
                            november, december
       september, october,
     );
   type packet is
     record
                : hours range;
       hours
                 : minutes range;
       minutes
               : seconds range;
       seconds
       hundredths : hundredths range;
                 : years_range;
       year
                  : months range;
       month
       day
                  : days range;
     end record;
   procedure set (
                  : hours_range;
       hours
       minutes
                : minutes range;
                 : seconds range;
       seconds
       hundredths : hundredths_range := 0
     );
   procedure get (
       hours
                 : out hours range;
                 : out minutes range;
       minutes
                 : out seconds range;
       seconds
       hundredths : out hundredths_range
     );
```

```
procedure set (
       month: months range;
       day : days range;
       year : years_range
     );
   procedure get (
       month : out months range;
       day
                : out days range;
       year : out years_range;
       days_name : out day of week
     );
   function get return packet;
   type time format is (
       long,
                        -- 03:00:00 AM
                      -- 03:00:00
       military,
                       -- 03:00
       short,
       none
     );
   type date_format is (
       long,
                        -- December 23, 1986
       month_day_year, -- 12/23/86
       day_month_year, -- 23-DEC-86
       none
     );
   function image (
                       : packet;
       time
       time_notation : time_format := long;
date_notation : date_format := long
     ) return string;
end time;
```

#### 18.1 Procedure Set

The overloaded set procedures are disambiguated by their parameters. The first version sets the time of day; the second version sets the date.

#### 18.1.1 Setting Time of Day

The first version of procedure set sets the system clock to a specified hour, minute, second, and hundredth of second.

```
procedure set (
   hours : hours_range;
   minutes : minutes_range;
   seconds : seconds_range;
   hundredths : hundredths_range := 0
);
```

This call corresponds to Interrupt 16#21#, function 16#2D#.

An example follows.

```
-- set the system time to 10:20:30.
--
with time;
use time;
procedure set_time is
begin
  set (hours => 10,
        minutes => 20,
        seconds => 30,
        Hundredths => 0);
end;
```

#### 18.1.2 Setting Date

The second version of procedure set sets the system clock to a specified date.

```
procedure set (
    month : months_range;
    day : days_range;
    year : years_range
);
```

This call corresponds to Interrupt 16#21#, function 16#2B#.

An example follows.

```
-- set the system date to July 27, 1986.
with time;
use time;
procedure set_date is
begin
   set (month => 7,
        day => 27,
        year => 1986);
end:
```

#### 18.2 Procedure Get

The overloaded get procedures are disambiguated by their parameters and/or return type. The first version obtains the time of day; the second version obtains the date; the third version returns a record containing both time and date.

#### 18.2.1 Getting Time of Day

The first version of procedure get obtains the time of day from the system clock.

```
procedure get (
   hours : out hours_range;
   minutes : out minutes_range;
   seconds : out seconds_range;
   hundredths : out hundredths_range
);
```

This call corresponds to Interrupt 16#21#, function 16#2C#.

#### 18.2.2 Getting the Date

The second version of procedure get obtains the date from the system clock.

```
procedure get (
    month : out months_range;
    day : out days_range;
    year : out years_range;
    days_name : out day_of_week
);
s call corresponds to Interrupt 16#21#, function
```

This call corresponds to Interrupt 16#21#, function 16#2A#.

An example follows.

```
-- get the current date and display the days name.

with text_io,
    time;

procedure display_day is
    mon : months_range;
    day : days_range;
    yr : years_range;
    today : day_of_week;

begin
    time.get (mon, day, yr, today);
    text_io.put_line (day_of_week'image (today));

end;
```

#### 18.2.3 Getting Both Date and Time

The third version of procedure get obtains both the date and time from the system clock.

```
function Get return Packet;
```

This call simply merges information from the date and time get procedures.

# 18.3 Function Image

Function image returns a string representation of an object of type packet in a specified format.

There is no corresponding DOS function.

For an example, refer to the previous section on function get.

Package time

Package tty provides operations on the console terminal display and keyboard.

This package can be used in place of text\_io under some circumstances. Package tty links in faster than text io and calls to the tty subprograms run faster.

#### **SPECIFICATION**

```
with common_display_types;
use common display types;
package tty is
   type shift status record is
      record
       insert
                   : boolean;
       caps lock
                   : boolean;
                   : boolean:
       num lock
       scroll lock : boolean;
       alt shift
                   : boolean;
       ctrl shift : boolean;
       left shift : boolean;
       right shift : boolean;
      end record;
   procedure clear screen;
   procedure put (char : character);
   procedure put (str : string);
   procedure put (
                      : row range;
                     : column range;
       column
       item
                     : string;
       underline
                     : boolean
                                  := false;
       reverse video : boolean
                                  := false;
       blink
                     : boolean
                                  := false;
                     : boolean
                                  := false
       intensity
     );
   procedure put (
                  : row_range;
       row -
       column
                  : column range;
       item
                  : string;
       foreground : color ;
       background : background color;
                  : boolean := false
     );
```

```
procedure put (
                          : row range;
          TOW
                          : column range;
          column
          item : character;
underline : boolean := false;
reverse_video : boolean := false;
                    : boolean := false;
          blink
          intensity : boolean := false
        );
     procedure put (
                : row_range;
          TOW
          column : column_ran
item : character;
                     : column range;
          foreground : color;
          background : background_color;
          blink : boolean := false
      procedure put_line (str : string);
      -- Keyboard Functions
      function shift status return shift_status_record;
      function char_ready return boolean;
      procedure get (
          scan code : out byte;
          char
                  : out character
        );
      function get (
          no echo : boolean := false;
          direct : boolean := false;
                   : boolean := false
          clear
        ) return character;
      procedure get (str : out string; last : out natural);
   end tty;
19.1 Procedure Clear Screen
Procedure clear_screen clears the currently active display.
   procedure clear screen;
This call corresponds to Interrupt 16#10#, function 16#06#.
19.2 Procedure Put
The overloaded put procedures provide various means of writing to the console display.
```

#### 19.2.1 **Put Character**

The first version of procedure put writes one character to the standard output device. procedure put (char : character);

This call corresponds to Interrupt 16#21#, function 16#02#.

#### 19.2.2 Put String

The second version of procedure put writes the specified string to the standard output device.

```
procedure put (str : string);
```

This call corresponds to Interrupt 16#21#, function 16#40#.

# 19.2.3 Put String, Non-Color Attributes

The third version of procedure put writes a string, item, with the specified non-color display attributes, to the specified cursor position in the currently selected page.

```
procedure put (
                    : row range;
    row
                    : column range;
    column
                   : string;
    item
                                := false;
                  : boolean
    underline
    reverse_video : boolean
                                := false;
                                 := false;
                   : boolean
    blink
                    : boolean
                                 := false
    intensity
  );
```

If the output string reaches the end of the bottommost screen line, the screen is automatically scrolled up one line. This may destroy any statically formatted screen display.

The initial row and column position must be valid for the current display mode. If the position is invalid, the results are unpredictable.

This call corresponds to Interrupt 16#10#, function 16#0E#.

# 19.2.4 Put String, Color Attributes

The fourth version of procedure put writes a string to the specified cursor position in the currently selected page with the specified color display attributes.

```
procedure put (
    row : row_range;
    column : column_range;
    item : string;
    foreground : color;
    background : background_color;
    blink : boolean := false
);
```

If the output string reaches the end of the bottommost line, the screen is automatically scrolled up one line. This may destroy any statically formatted screen display.

The initial row and column position must be valid for the current display mode. If the position is invalid, the results are unpredictable.

This call corresponds to Interrupt 16#10#, function 16#0E#.

# 19.2.5 Put Character, Non-Color Attributes

The fifth version of procedure put writes a character to the specified cursor position in the currently selected page with the specified non-color display attributes.

```
procedure put (
    row : row_range;
    column : column_range;
    item : character;
    underline : boolean := false;
    reverse_video : boolean := false;
    blink : boolean := false;
    intensity : boolean := false;
};
```

If the output character reaches the end of the bottommost line, the screen is automatically scrolled up one line. This may destroy any statically formatted screen display.

The initial row and column position must be valid for the current display mode. If the position is invalid, the results are unpredictable.

This call corresponds to Interrupt 16#10#, function 16#0E#.

## 19.2.6 Put Character, Color Attributes

The sixth version of procedure put writes a character to the specified cursor position in the currently selected page with the specified color display attributes.

```
procedure put (
   row : row_range;
   column : column_range;
   item : character;
   foreground : color;
   background : background_color;
   blink : boolean := false
);
```

If the output character reaches the end of the bottommost line, the screen is automatically scrolled up one line. This may destroy any statically formatted screen display.

The initial row and column position must be valid for the current display mode. If the position is invalid, the results are unpredictable.

This call corresponds to Interrupt 16#10#, function 16#0E#.

# 19.3 Procedure Put\_Line

Procedure **put\_line** writes the specified string to the standard output device followed by a carriage—return, line—feed sequence.

```
procedure put_line (str : string);
```

This call corresponds to Interrupt 16#21, function 16#40#.

# 19.4 Function Shift\_Status

The function shift\_status returns the present shift status of the keyboard.

```
function shift_status return shift_status_record;
```

This status record corresponds to the ROM BIOS flag at address 0000:0417H.

This call corresponds to Interrupt 16#16#, function 16#02#.

# 19.5 Function Char\_Ready

The function char\_ready determines if a character is ready to be read.

```
function char_ready return boolean;
```

This call corresponds to Interrupt 16#16#, function 16#01#.

# 19.6 Subprogram Get

The overloaded get subprograms provide various means of reading from the console keyboard.

#### 19.6.1 Raw Get

The first version of get obtains the scan code and raw character from the keyboard.

```
procedure get (
    scan_code : out byte;
    char : out character
);
```

This call corresponds to Interrupt 16#16#, function 16#00#.

#### 19.6.2 Get Character

The second version of get returns a character from the standard input device by the specified method.

```
function get (
   no_echo : boolean := false;
   direct : boolean := false;
   clear : boolean := false
) return character;
```

If the clear flag is set then the input buffer is first cleared before the next character is input.

If the no\_echo and direct flags are both set, then Interrupt 16#21#, function 16#07# is used. Characters are read without being echoed to the standard output device. No special action is taken if a Control-C (^C) is detected.

If the no\_echo flag alone is set then Interrupt 16#21#, function 16#08# is used. Characters are read without being echoed to the standard output device.

If the direct flag is set then Interrupt 16#21#, function 16#06# is used. No special action is taken if a Control-C (^C) is detected. If no character is ready for input then a null character (ASCII NUL) is returned.

# 19.6.3 Get String

The third version of get reads a string from the standard input device.

```
procedure get (str : out string; last : out natural);
```

This call performs a buffered read until a carriage return is detected.

Parameter last is the index position of the last valid character in the returned string.

This call corresponds to Interrupt 16#21#, function 16#0A#.

Package tty

Package video provides various output and control functions for the Monochrome, Color Graphics, and Extended Graphics Adapter Cards.

The Color/Graphics Adapter has eight display pages (0.. 7) in 40 column by 25 line text modes and four display pages in the 80 column by 25 line text modes. The Monochrome adapter has only one display page. Specifying an invalid display page (i.e. a display page that is not defined for the current text display mode) has unpredictable results.

Some of the overloaded subprograms are disambiguated by the type of character that they accept: either standard.character or common\_display\_types.extended\_ascii. Be aware that the Ada pre-defined type character supports only seven-bit ASCII characters. If it is necessary to manipulate all eight bits of a byte, then pragma suppress should be used when using objects of type character (to avoid the exception constraint\_error), or the type extended\_ascii should be used.

### **SPECIFICATION**

```
with common display types;
use common display_types;
package video is
   type video mode is
       text40 bw,
       text40 co,
       text80 bw,
       text80 co,
       graphic 4 color,
       graphic_4_grey,
       graphic bw,
       text80 bw ma,
       graphic color high,
       graphic med_ega,
       graphic high_ega,
       graphic extra_ega,
       color extra ega
   type pixel_value is
       record
                                     := false;
                    : boolean
       pixel_color : graphic_color := background;
       end record;
   procedure set (mode : video_mode);
    procedure get mode (
        width : out byte;
        mode : out video mode;
        page : out display_page
      );
```

```
procedure get_light_pen (
     triggered : out boolean;
    pixel_row : out integer;
    pixel_col : out integer;
    char_row : out integer;
    char_col : out integer
  );
procedure set_active (page : display_page);
-- Windowing functions
procedure scroll_up (
    number_of_lines : byte;
    upper_row
                : row_range;
    left_column
                  : column_range;
    lower row
                  : row range;
    right column : column range;
    filler_attribute: display_attribute := (
        foreground => white,
        background => black,
        blink
                 => false
      )
  );
procedure scroll down (
    number_of_lines : byte;
    upper row
                   : row_range;
    left column
                   : column_range;
    lower row
                    : row range;
    right column
                   : column range;
    filler_attribute : display_attribute := (
        foreground => white,
       background => black,
       blink => false
      )
  );
procedure clear screen;
procedure read char (
            : out extended ascii;
    attribute : out display_attribute;
   page
             : display_page := 0
  );
```

```
procedure write_char (
              : character;
    repeat_count : natural;
    attribute : display_attribute := (
        foreground => white,
        background => black,
        blink => false
      );
               : display_page := 0
    page
  );
procedure write_char (
                 : extended_ascii;
    item
    repeat_count : natural;
    attribute : display_attribute := (
        foreground => white,
        background => black,
        blink => false
                : display_page := 0
     page
   );
 procedure write_char (
                 : character;
     repeat_count : natural;
                 : display page
     page
   );
 procedure write_char (
                 : extended_ascii;
     item
     repeat_count : natural;
                 : display page
     page
   );
 procedure set_color_palette (to_color : color);
 procedure write_tty (item : character);
 procedure write_tty (item : extended_ascii);
  -- Graphics Mode Only
  procedure set_color_palette (
      to_palette : color_palette
    );
  procedure write_tty (
                 : character;
      which_color : pixel_value
    );
```

```
procedure write pixel (
           pixel_color : pixel_value;
            pixel_row : natural;
           pixel_column : natural
         );
       function read_pixel (
           pixel_row : natural;
           pixel_column : natural
         ) return pixel_value;
      procedure write_graphic_char (
                         : character;
           attribute
                         : pixel value;
           repeat count : natural
         );
   end Video;
20.1 Type Video Mode
The enumeration elements of video_mode are described in Figure 20.1.
20.2 Procedure Set
Procedure set sets the current video display mode.
   procedure set (mode : video mode);
This call corresponds to Interrupt 16#10#, function 16#00#.
An example follows.
   -- set the video mode to 25x40 color text
   video.set (mode => text40 co);
20.3 Procedure Get Mode
Procedure get_mode obtains the current display mode, screen width, and page.
   procedure get mode (
       width : out byte;
       mode : out video mode;
       page : out display page
     );
This call corresponds to Interrupt 16#10#, function 16#0F#.
20.4 Procedure Get_Light_Pen
Procedure get_light_pen obtains the light pen's current status and position.
  procedure get_light pen (
       triggered : out boolean;
       pixel_row : out integer;
       pixel col : out integer;
       char_row : out integer;
       char col : out integer
     );
```

This call corresponds to Interrupt 16#10#, function 16#04#.

# 20.5 Procedure Set\_Active

Procedure set\_active sets the active video display page.

```
procedure set_active (page : display_page);
```

Valid display pages are:

- 0.. 7 for text40\_bw (mode 0), text40\_co (mode 1) Color/Graphics Adapter
- 0.. 3 for text80\_bw (mode 2), text80\_co (mode 3) Color/Graphics Adapter
- 0.. 7 for text80\_bw (mode 2), text80\_co (mode 3) Extended Adapter

Setting an invalid page for the current mode has unpredictable results.

Procedure set\_active does not apply to a Monochrome display.

This call corresponds to Interrupt 16#10#, function 16#05#.

An example of using package video with procedure set\_active follows.

```
-- set the active display page to page 1.
--
video.set_active (page => 1);
```

# 20.6 Procedure Scroll\_Up

Procedure scroll\_up scrolls a specified window up and initializes the new lines with the filler\_attribute.

```
procedure scroll_up (
   number_of_lines : byte;
   upper_row : row_range;
   left_column : column_range;
   lower_row : row_range;
   right_column : column_range;
   filler_attribute : display_attribute := (
        foreground => white,
        background => black,
        blink => false
   )
);
```

If number\_of\_lines is zero then the entire window is blanked and initialized with the filler\_attribute. Lines scrolled above the top of the window are lost.

Scrolling is only valid for the currently selected display page.

Scrolling invalid windows has unpredictable results.

This call corresponds to Interrupt 16#10#, function 16#06#.

An example of using package video with procedure scroll\_up follows.

```
-- define a window with corners 5,10 and 15,30 (row,col)
-- and scroll it up 4 lines filling in the bottom lines
-- with a blue background.
--
with video, common_display_types;
use common_display_types;
procedure make_window is
```

```
fill_attr : display_attribute;
begin
   video.clear screen;
   fill_attr.background := blue;
   video.scroll_up (
       number_of_lines => 4,
       upper row
                        => 5,
       left_column
                        => 10,
       lower row
                        => 15,
       right column
                        => 30,
       filler_attribute => fill_attr
     );
end;
```

# 20.7 Procedure Scroll Down

Procedure scroll\_down scrolls a specified window down and initializes the new lines with the filler\_attribute.

```
procedure scroll down (
   number_of_lines : byte;
   upper row
                  : row range;
   left column
                 : column_range;
   lower row
                   : row range;
   right column
                   : column_range;
   filler_attribute : display_attribute := (
       foreground => white,
       background => black,
       blink
             => false
     )
 );
```

If number\_of\_lines is zero then the entire window is blanked and initialized with the filler\_attribute.

Scrolling is only valid for the currently selected display page.

Lines scrolled below the bottom of the window are lost.

Scrolling invalid windows has unpredictable results.

This call corresponds to Interrupt 16#10#, function 16#07#.

# 20.8 Procedure Clear Screen

Procedure clear\_screen clears the currently selected display page.

```
procedure clear screen;
```

This call corresponds to Interrupt 16#10#, function 16#06#.

An example of using package video with procedure clear\_screen follows.

```
-- erase the screen.
--
video.clear_screen;
```

# 20.9 Procedure Read\_Char

Procedure read\_char allows eight—bit characters (bytes) to be read. Procedure read\_char obtains the character and its attribute from the current cursor position in the specified display page.

```
procedure read_char (
       item : out extended_ascii;
       attribute : out display_attribute;
             : display_page := 0
     );
This call corresponds to Interrupt 16#10#, function 16#08#.
An example of using package video with procedure read_char follows.
   -- get the character and attribute from display
   -- page zero.
   declare
       char : extended_ascii;
       attr : display_attribute;
   begin
       video.read_char (item => char, attribute => attr);
20.10 Procedure Write_Char
The overloaded write_char procedures are distinguished by the presence or absence of the attribute
parameter and by the type of character written.
 20.10.1 Write With Attributes
 The first and second versions of procedure write_char write a character and its attribute at the current
 cursor position in the specified display page.
 Seven-Bit ASCII
 The first version of procedure write_char writes a seven-bit ASCII character.
    procedure write_char (
                        : character;
         item
         repeat_count : natural;
                      : display_attribute := (
         attribute
             foreground => white,
             background => black,
             blink => false
            );
                        : display_page := 0
         page
       );
 The cursor position is not updated.
 The results are unpredictable if the repeat_count causes characters to be written past the right side of the
  This call corresponds to Interrupt 16#10#, function 16#09#.
  An example which uses package video with procedure write_char follows.
```

-- Write a string of 10 happy faces (ASCII ordinal 1) at

-- the current cursor position.

procedure write happy is

with video;

begin

```
video.write_char (
        item
                      => character'val (1),
        repeat_count => 10
       );
   end:
Eight-Bit Characters
The second version of procedure write_char writes an eight-bit character.
  procedure write_char (
      item
                    : extended ascii;
      repeat_count : natural;
      attribute
                   : display_attribute := (
          foreground => white,
         background => black,
         blink => false
        );
                : display page := 0
      page
    );
```

The cursor position is not updated.

The results are unpredictable if the *repeat\_count* causes characters to be written past the right side of the screen.

This call corresponds to Interrupt 16#10#, function 16#09#.

### 20.10.2 Write With Previous Attributes

The third and fourth versions of procedure write\_char write the character at the current cursor position in the specified display page with the already-present attributes.

#### Seven-Bit ASCII

The third version of procedure write\_char writes a seven-bit ASCII character.

```
procedure write_char (
   item : character;
   repeat_count : natural;
   page : display_page
);
```

The character receives the attribute of the previous character at that position.

The cursor position is not updated.

The results are unpredictable if the *repeat\_count* causes characters to be written past the right side of the screen.

This call corresponds to Interrupt 16#10#, function 16#0A#.

#### Eight-Bit Characters

The fourth version of procedure write\_char writes eight-bit characters.

The character receives the attribute of the previous character at that position.

The cursor position is not updated.

The results are unpredictable if the repeat\_count causes characters to be written past the right side of the screen.

This call corresponds to Interrupt 16#10#, function 16#0A#.

# 20.11 Procedure Set\_Color\_Palette

The overloaded set\_color\_palette procedures are distinguished by a color parameter appropriate to the display device.

# 20.11.1 Text or Graphics

The first version of procedure set\_color\_palette sets the background and border color in graphics modes, or sets just the border color in text modes.

```
procedure set_color_palette (to_color
                                       : color);
```

This call corresponds to Interrupt 16#10#, function 16#0B#.

An example which uses package video with procedure set\_color\_palatte follows.

```
-- set the text modes border color to blue.
video.set_color_palette (to_color => blue);
    -- Note: must already be in a text mode.
```

#### 20.11.2 CGA

The second version of procedure set\_color\_palette sets the specified palette for the Color/Graphics Adapter.

```
procedure set_color_palette (to_palette : color_palette);
```

This call is valid only in Graphic\_4\_Color mode.

This table indicates what colors are presented:

Color_Pallette 0	Graphic_Color	
	Background Color1 Color2 Color3	= Same as current = Green = Red = Brown
1	Background Color1 Color2 Color3	= Same as current = Cyan = Magenta = White

This call corresponds to Interrupt 16#10#, function 16#0B#.

An example follows.

```
-- set the color palette to palette 1.
video.set_color_palette (to_palette => 1);
```

# 20.12 Procedure Write\_Tty

The overloaded write\_tty procedures are disambiguated by parameters appropriate to the output device.

# 20.12.1 Write To Current Display Page

The first and second versions of procedure write\_tty write a character to the currently active display page and updates the cursor position.

#### Seven-Bit ASCII

The first version of procedure write\_tty writes seven-bit ASCII characters.

```
procedure write_tty (item : character);
```

This call should be made in text modes only.

Appropriate action is taken when these special characters are written:

- bell (ASCII BEL)
- linefeed (ASCII LF)
- carriage return (ASCII CR)
- backspace (ASCII BS)

The display attribute is taken from the previous character at that location.

If the cursor reaches the end of the bottommost screen line, the screen automatically scrolls up and the cursor is positioned at the "next" line.

This call corresponds to Interrupt 16#10#, function 16#0E#.

An example which uses package video with procedure write-tty follows.

```
-- write out the the string "this is a test"

with video;
procedure write_tty_test is

str : string (1 .. 14) := "this is a test";
begin

for i in str'range loop

video.write_tty (item => str (i));
end loop;
end;
```

#### Eight-Bit Characters

The second version of procedure write\_tty writes eight-bit characters.

```
procedure write_tty (item : extended_ascii);
```

Appropriate action is taken when these special characters are written:

- bell (ASCII BEL)
- linefeed (ASCII LF)
- carriage return (ASCII CR)

### • backspace (ASCII BS)

The display attribute is taken from the previous character at that location.

If the cursor reaches the end of the bottommost screen line, the screen automatically scrolls up and the cursor is positioned at the "next" line.

This call corresponds to Interrupt 16#10#, function 16#0E#.

An example follows.

```
-- write out the the string "this is a test"

with video;
procedure write_tty_test2 is
    str : string (1 .. 14) := "this is a test";
begin
    for i in str'range loop
        video.write_tty (item => character'pos (str (i)));
    end loop;
end;
```

### 20.12.2 Write Graphic Character

The third version of procedure write\_tty writes a graphic character with the specified attribute.

```
procedure write_tty (
    item : character;
    which_color : pixel_value
);
```

The results are unpredictable if the current display mode is not a graphic mode.

If the cursor reaches the end of the bottommost screen line, the screen automatically scrolls up and the cursor is positioned at the "next" line.

This call corresponds to Interrupt 16#10#, function 16#0E#.

# 20.13 Procedure Write\_Pixel

Procedure write\_pixel turns on a pixel at the specified location.

```
procedure write_pixel (
    pixel_color : pixel_value;
    pixel_row : natural;
    pixel_column : natural
);
```

Valid display locations depend on the current graphic mode. The results are unpredictable for invalid coordinates.

This call corresponds to Interrupt 16#10#, function 16#0C#.

# 20.14 Function Read\_Pixel

Function read\_pixel obtains the pixel\_value for the specified position.

```
function read_pixel (
    pixel_row : natural;
    pixel_column : natural
) return pixel_value;
```

Valid display locations depend on the current graphic mode.

This call corresponds to Interrupt 16#10#, function 16#0C#.

# 20.15 Procedure Write\_Graphic\_Char

Procedure write\_graphic\_char writes a graphic character at the current cursor location.

```
procedure write_graphic_char (
   item : character;
   attribute : pixel_value;
   repeat_count : natural
);
```

The cursor position is not updated.

This call is valid only in graphics modes.

The results are unpredictable if the *repeat\_count* causes characters to be written past the right side of the screen.

This call corresponds to Interrupt 16#10#, function 16#0A#.

An example which uses package video with procedure write\_graphic\_char follows.

```
-- write the 10 graphic '?' characters,
-- wait 2 seconds then erase it.
-- (graphics mode is assumed)
with video, common_display_types;
use video, common_display_types;
procedure graphic test is
   attr : pixel value;
begin
   attr.pixel color := color1;
   write_graphic_char (
                   => '?',
       item
                   => attr,
       attribute
       repeat count => 10
     );
   delay 2.0;
   attr.xor_bit := true;
   write graphic char (
       item
                   => '?',
       attribute
                   => attr,
       repeat_count => 10
     );
end;
```

Video_Mode	mode #	description
text40_bw	00H	25x40 B/W text, Color Graphics Adapter (CGA)
text40 co	01H	25x40 Color text (CGA)
text80 bw	02H	25x80 B/W text (CGA)
text80_co	03H	25x80 Color text (CGA)
graphic_4_color	04H	200x320 4 color graphics (CGA)
graphic 4_grey	05H	200x320 4 grey (CGA)
graphic_bw	06H	200x640 2 color (CGA)
text80_bw_ma	07H	25x80 B/W text, Monochrome Adapter
graphic_color_high	0AH	200x640 color graphics, Ex- tended Adapter (EGA)
graphic_med_ega	0DH	200x320 16 color graphics (EGA)
graphic_high_ega	0EH	200x640 16 color graphics (EGA)
graphic_extra_ega	0FH	350x640 monochrome graphics (EGA)
color_extra_ega	10H	350x640 four color or 16-color graphics (EGA)

Figure 20.1Video Modes

and the second s	Market and September 1991	
The second second	Ashor	92. <b>188</b> (1.7.2)
Section of the sectio	Company of the Company	The second secon
And the parties	MAI .	शर्व है। अक्ष
Service Constitution	الله المال المراسد العربية والمراسد العربية المراسة العربية المراسة المراسة المراسة المراسة المراسة المراسة ال المراسد المراسة	and the second s
	1110	03 Av. (20)
	HSO	wei die
	HO	The state of the s
ACCA	140	78 W 1 7 1 W 1 9
Brown Bar Bar	The second secon	en ann a sapaag
Addition the party of		
The World Deal Wish open	ATO I	And the second state of the second se
of help A		A STATE OF THE STA
The state of the s		Pyrinceton agests
7 10 15 7 10 16 (R 10 00). 4 13	TO:	The state of the s
Reform the Affect of the	The second secon	Age Asti Caper
्रिक्त विशेषक (१०००) स्थापित हो। १८८ मार्ग	The service of the se	िक्षेत्र । स्थाप्तक । अन्य सुक्षाक,
THE RESIDENCE PROPERTY	ere	and an extended the second of

# **Symbols**

- \*.aar files, 4
- \*.int files, 4
- \*.lib files, 4

# A

absolute\_disk. See package absolute\_disk
Ada for Programmers, 2
ada.lib, 3
allocate, procedure, 58—59
amount of memory installed, 60
ANSI/MIL-STD-1815A (LRM), 2
ASCII BEL, 90
ASCII BS, 90, 91
ASCII CR, 90
ASCII LF, 90

# B

bin directory, 4

border color
graphics, 89
text, 89

box style, selecting, 11

box, package. See package box
boxdemo.ada file, 4
break\_status, function, 64

buffers, flushing, 25

byte input, 61—62

# C

calls to interrupt vectors, 55—56 cdup\_filehandle, procedure, 52 CGA (color graphics adapter), 89

change\_to, function, 22-23 char\_ready, function, 79 character set for IBM PC, 13 characters for drawing boxes, 11-14 clear\_screen, procedure, 76, 86 clearing the screen, 76, 86 close, function, 45-46 color graphics card, 81 color, setting palette, 89 color\_extra\_ega, 93 common\_display\_types. See package common\_display\_types common\_display\_types.extended\_ascii, 81 cooked, function, 46 create, procedure, 41-42 create\_new, procedure, 44 create\_temporary, procedure, 42-43 current directory with procedure create\_temporary, 43 current\_name, function, 23-24 cursor making invisible, 18 manipulating, 17-20 moving, 18 moving down, 19 moving left, 19 moving right, 19-20 moving up, 19 positioning, 18-19 cursor, package. See package cursor

# D

date
getting, 72
setting, 71
decreasing memory block, 59
delete, function, 48
directories
changing, 22—23
making, 21—22

obtaining pathname, 23-24 renaming, 48-49 removing, 22 writing to, 47 directory, package. See package directory files, creating, 41—42 disk drive find\_first, procedure, 52-53 obtaining disk information, 27-28 find\_next, procedure, 53—54 obtaining information about, 26 requesting ID, 26 flags component, 56 setting default, 26 function break\_status, 64 verifying read-after-write, 26, 27 function change—to, 22—23 disk, package. See package disk function char\_ready, 79 disk\_types, package, 29 function close, 45-46 display mode, 84 function cooked, 46 DOS Environment Library, installing, 3-6 function current\_name, 23--24 dos, function, 67 function delete, 48 dosenv directory, 3, 4 function dos, 67 down, procedure, 19 function execute, 65 draw, procedure, 11-14 function get\_default, 26 dup\_filehandle, 52 function image, 73 function in\_byte, 61 E function in\_word, 61 function installed, 60 e:\stuff directory, 3 function list, 31 eight-bit characters, 88, 90 function make, 21, 58 eight-bit characters, reading, 86 function msdos, 65—66 envdisp, 5 function raw, 46 envdisp.ada file, 4 function read\_pixel, 91—92 equipment, package, 31 function release, 59 errors, package. See package errors function remove, 22 exception constraint\_error, 81 function rename, 48-49 exception incorrect\_dos\_version, 67 function resident quit, 64 execute, function, 65 function segment\_prefix, 64 extended graphics adapter card, 81 function set\_attributes, 51 extended status information, 33 function set\_default, 26 function set\_file\_time, 51 function shift\_status, 78 function used, 60 file\_io, package. See package file\_io function verification\_is\_on, 27 files closing, 45—46 G creating, 44 deleting, 48

opening, 45 reading from, 46

get string, 79

get, procedure, 71-72

	interrupt 16#10#
get, subprogram, 79—80	function 16#00#, 84
get_allocation_info, procedure, 26	function 16#01#, 18
get_attributes, procedure, 50-51	function 16#02#, 18, 19 function 16#03#, 19
get_default, function, 26	function 16#04#, 85
get_environment_variable, procedure, 66	function 16#05#, 85
get_extended_info, procedure, 34	function 16#06#, 76, 85—86 function 16#07#, 86
get_file_time, procedure, 51—52	function 16#09#, 87, 88
get_free_space, procedure, 27—28	function 16#0A#, 88, 89, 92
	function 16#0B#, 89 function 16#0C#, 91, 92
get_light_pen, procedure, 84—85	function 16#0E#, 77, 78, 90, 91
get_mode, procedure, 84	function 16#0F#, 84
get_position, procedure, 18—19	interrupt 16#11#, 31
get_return_code, procedure, 64	interrupt 16#16#
graphic characters, writing, 91, 92—93	function 16#00#, 79
graphic_4_color, 93	function 16#02#, 78, 79
graphic_4_color mode, 89	interrupt 16#21#
graphic_4_grey, 93	function 16#02#, 76 function 16#06#, 79
graphic_bw, 93	function 16#07#, 79
	function 16#0A#, 79
graphic_color_high, 93	function 16#0D#, 25
graphic_extra_ega, 93	function 16#0E#, 26 function 16#12#, 60
graphic_high_ega, 93	function 16#19#, 26
graphic_med_ega, 93	function 16#1C#, 26
Series 7 2 0 1	function 16#2A#, 72
	function 16#2B#, 71
H	function 16#2C#, 72 function 16#2D#, 70
· · · · · · · · · · · · · · · · · · ·	function 16#2E#, 76
host operating system version number, 67	function 16#31#, 64
	function 16#33#, 64
1	function 16#36#, 27
	function 16#3C#, 41
	function 16#3D#, 45
image, function, 73	function 16#3E#, 46
IN instruction, 61	function 16#3F#, 47
in_byte, function, 61	function 16#40#, 48, 77, 78
_ •	function 16#41#, 48
in_word, function, 61	function 16#42#, 49 function 16#43#, 50, 51
incorrect_dos_version, exception, 67	function 16#44#, 46
increasing memory block, 59—60	function 16#45#, 52
inhibit, procedure, 18	function 16#46#, 52 function 16#48#, 58
install directory, 4	function 16#49#, 59
INSTALL program, 3	function 16#4A#, 60
installed, function, 60	function 16#4B#, 65 function 16#4C#, 64
installing the DOS Environment Library, 3—6	function 16#4D#, 64
installing the DOS Elivironilient Liolary, 3220	Thirties The Land A.

function 16#4E#, 53 open, procedure, 45 function 16#54#, 27 **OUT** instruction, 62 function 16#56#, 49 function 16#57#, 51 out\_byte, procedure, 62 function 16#59#, 35 out\_word, procedure, 62 function 16#5A#, 42, 43 function 16#5B#, 44 interrupt 16#25#, 7-8 interrupt 16#26#, 8 package absolute\_disk, 7—8 interrupt vectors, calls to, 55 procedure read, 7-8 procedure write, 8 interrupt, package. See package interrupt package box, 9-14 procedure draw, 11-14 type part, 10 type simple\_kind, 11 type user\_definition, 11 left, procedure, 19 package common\_display\_types, 15-16 light pen position, 84—85 package cursor, 17-20 status, 84—85 procedure down, 19 procedure get\_position, 18--19 Inlib command, 3 procedure inhibit, 18 procedure left, 19 procedure move, 18 procedure right, 19-20 procedure set\_size, 17-18 make, function, 21, 58 procedure up, 19 memory block, releasing, 59 package directory, 21-24 function change\_to, 22-23 memory, allocating and deallocating, 57-60 function current\_name, 23-24 memory, package. See package memory function make, 21-22 Microsoft MS-DOS Operating System Programmer's function remove, 22 Reference Guide, 2 package disk, 25-28 modify, procedure, 59-60 function get\_default, 26 function set\_default, 26 monochrome graphics card, 81 function verification\_is\_on, 27 move, procedure, 18 procedure get\_allocation\_info, 26 procedure get\_free\_space, 27-28 move\_file\_pointer, procedure, 49-50 procedure reset, 25 msdos, function, 65-66 procedure set\_verification, 26 package disk\_types, 29 package equipment, 31—32 N function list, 31 newlib command, 4 package errors, 33-36 procedure get\_extended\_info, 34 newlib.bat file, 4 package file\_io, 37—54 function close, 45-46 function cooked, 46 function delete, 48 function raw, 46 object type packet, 73

offset, 58

function rename, 48-49

function set\_attributes, 51

function set_file_time, 51	package video, 81—94
procedure cdup_filehandle, 52	enumeration elements for video_mode, 93
procedure create, 41—42	function read_pixel, 91—92
procedure create_new, 44	procedure clear_screen, 86
procedure create_temporary, 42—43	procedure get_light_pen, 84—85
procedure dup_filehandle, 52	procedure get_mode, 84
procedure find_first, 52—53	procedure read_char, 86—87
procedure find_next, 53—54	procedure scroll_down, 86
procedure get_attributes, 50—51	procedure scroll_up, 85—86
procedure get_file_time, 51—52	procedure set, 84
procedure move_file_pointer, 49—50	procedure set_active, 85 procedure set_color_palette, 89
procedure open, 45	procedure write_char, 87—89
procedure read, 46—47	procedure write_graphic_char, 92—93
procedure write, 47—48	procedure write_pixel, 91
package interrupt, 55—56	procedure write_tty, 90
procedure vector, 55—56	type video_mode, 84, 93
•	
package memory, 57—60	paclib\ada.lib, 3
function installed, 60	part, type, 10
function make, 58	parts of box, creating, 10, 11
function release, 59 function used, 60	
procedure allocate, 58—59	pathname with procedure create_temporary, 42-43
procedure modify, 59—60	PC-DOS version number, 67
procedure split, 58	port, package, 61—62
	•
package port, 61—62	pragma suppress, 81
function in_byte, 61	procedure allocate, 58—59
function in_word, 61 procedure out_byte, 62	procedure cdup_filehandle, 52
procedure out_word, 62	procedure clear_screen, 76, 86
· · · · · · · · · · · · · · · · · · ·	
package program_control, 63—66	procedure create, 41—42
function break_status, 64	procedure create_new, 44
function execute, 65	procedure create_temporary, 42—43
function msdos, 65—66	
function resident_quit, 64 function segment_prefix, 64	procedure down, 19
procedure get_environment_variable, 66	procedure draw, 1114
procedure quit, 64	pre-defined box types, 11—12
procedure set_break_status, 64	selected characters for box parts, 13—14
package revision, 67—68	procedure dup_filehandle, 52
exception incorrect_dos_version, 67	procedure find_first, 52—53
function dos, 67	procedure find_next, 53—54
package time, 69—74	procedure get, 71
function image, 73	procedure get_allocation_info, 26
procedure get, 71—72 procedure set, 70	procedure get_attributes, 50—51
package tty, 75	procedure get_environment_variable, 66
function char_ready, 79	•
function shift_status, 78	procedure get_extended_info, 34
procedure clear_screen, 76	procedure get_file_time, 51—52
procedure put, 76—78	procedure get_free_space, 27—28
procedure put_line, 78 subprogram get, 79—80	procedure get_light_pen, 8485
2007707388 450U	France Orang me

procedure get\_mode, 84 procedure get\_position, 18—19 procedure get\_return\_code, 64 procedure inhibit, 18 procedure left, 19 procedure modify, 59-60 procedure move, 18 procedure move\_file\_pointer, 49--50 procedure open, 45 procedure out\_byte, 62 procedure out word, 62 procedure put, 76-78 procedure put line, 78 procedure quit, 64 procedure read, 7—8, 46—47 procedure read\_char, 86-87 procedure reset, 25 procedure right, 19-20 procedure scroll\_down, 86 procedure scroll\_up, 85—86 procedure set, 70, 84 procedure set\_active, 85 procedure set\_break\_status, 64 procedure set\_color\_palette, 89 procedure set\_size, 17-18 procedure set\_verification, 26 procedure split, 58 procedure up, 19 procedure vector, 55—56 procedure write, 8, 47-48 procedure write\_char, 87-89 procedure write\_graphic\_char, 92—93 procedure write\_pixel, 91 procedure write\_tty, 90 procedure write\_tty\_test, 90 program size at startup, 60 program\_control, package. See package program\_control put character, 76 put character, color attributes, 78

put character, non-color attributes, 77
put string, 77
put string, color attributes, 77
put string, non-color attributes, 77
put\_line, procedure, 78

### Q

quit, procedure, 64

### R

raw get, 79 raw, function, 46 read, procedure, 7-8, 46-47 read\_char, procedure, 86-87 read\_pixel, function, 91—92 reading eight-bit characters, 86 registers, 55—56 release, function, 59 releasing memory block, 59 remove, function, 22 rename, function, 48-49 reset, procedure, 25 resident\_quit, function, 64 return code, getting, 64 revision.incorrect dos version, 43 revision, package. See package revision right, procedure, 19-20

# S

screen width, 84
screen, clearing, 86
scroll\_down, procedure, 86
scroll\_up, procedure, 85—86
scrolling
down, 86
up, 85—86
segment, 58
segment\_prefix, function, 64
set procedure, 70

set, procedure, 84 set\_active, procedure, 85 set\_attributes, function, 51 set\_break\_status, procedure, 64 set\_color\_palette, procedure, 89 set\_default, function, 26 set\_file\_time, function, 51 set\_size, procedure, 17 set\_verification, procedure, 26 seven-bit ascii, 87, 88, 90 shift\_status, function, 78 simple\_kind, type, 11 spilt, procedure, 58 standard.character, 81 status declarations, 33 subprogram get, 79—80

### T

temporary files, creating, 42—43
terminating the calling program, 64
test directory, 4
text\_io, 75
text40\_bw, 93
text40\_co, 93
text80\_bw\_ma, 93
text80\_bw\_ma, 93
text80\_co, 93
time of day
getting, 71—72
setting, 70—71
time, package. See package time
tty, package. See package tty

type character, 81 type extended\_ascii, 81 type part, 10 type simple\_kind, 11 type user\_definition, 11 type video\_mode, 84, 93

### U

up, procedure, 19
used, function, 60
user\_definition, type, 11

### V

vector, procedure, 55—56
verification\_is\_on, function, 27
video, package. See package video
video\_mode enumeration elements, 93
video\_mode, type, 84, 93

# W

word input, 61—62
write, procedure, 8, 47—48
write\_char, procedure, 87—89
write\_graphic\_char, procedure, 92—93
write\_pixel, procedure, 91
write\_tty, procedure, 90
write\_tty\_test, procedure, 90
writing characters, 90
writing to console display, 76